# CoCoNuts!

May 2007 Volume 3 Issue 2

THE LIFE AND TIMES OF THE COLOR COMPUTER


Wow what a year this has been so far. I want to thank everyone for making this dream of mine come true. I can only hope that my dad is smiling down on me from heaven. I know if we would have been able to continue through life together he would have wanted me to be part of this community back in those times.

I think I have done a pretty good job of bringing most of us closer together with something we all love. We all love a good magazine about our favorite machines. I have come to learn that all CoCoNuts seem to have three things in common: Astronomy, Star-Trek, and Dr. Pepper.

I'm sure you'll note the big change in format with this issue of CoCoNuts! Steve (6809er) Bjork has stepped in as our art director and typesetter. He has given us a beautifull new look that is also easier to read. I'm sure we will see more of Steve Bjork's work in future issues too.

So if you guys behave yourselves and keep getting submissions in on time I might just add a couple articles that review our favorite episodes of Star-Trek, or add in some pictures of vintage Dr. Pepper cans! I hope you enjoy this issue of CoCoNutz! E-Zine.

Mary Kramer, Executive Editor

## What's inside ...




### Our Staff

| | |
|---|---|
| Executive Editor | Mary Kramer |
| Chief consultant | Bob Devries (dad) |
| Publishing | Roger Taylor |
| Senior Writer | Robert Gault |
| Staff Writers | Steve (6809er) Bjork<br>Cris (CaptCPU) Egger<br>Allen (OS-9er) Huffman<br>Richard Ivey<br>John (Sock Master) Kowalski<br>Brian (Briza) Palmer |
| Art Director & Typesettting | Steve (6809er) Bjork |
| Photos | Allen (OS-9er) Huffman<br>Steve (6809er) Bjork |

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

# Getting to Know: Steve Bjork!

Steve has been missing from the CoCo scene for a long time. When someone challenged me to try to get him back I immediately sent out the email. After waiting for a short period of time I pounced on him in the CoCo3.com chat one evening.

***"Steve's wonderful personality quickly won me over."***

The ball started to roll. Steve and I agreed to meet at the fest and do a video taped interview. I considered this a great honor and was a bit nervous. Steve's wonderful personality quickly won me over. He is a very polite person and was an absolute pleasure to talk with. I am so proud to offer this interview for this issue.

Now Steve is extremely active in our CoCo community once more. Welcome home Steve! This is the transcript from that video taped interview. I want to thank him for this chance to do this, and also all the help he has given me on this issue of the E-Zine.

Mary: We are here with Steve Bjork. (Hi Steve)

Steve: Hi Everybody!

Mary: We will start off with "Where do you live?"

Steve: First of all, I'm going to pronounce my name so everyone gets it correct, it's "Be-York". It's Scandinavian; my father is 100% Swedish and comes from Minnesota. (Bjork is a common name back there.)

Anyway, I'm born and raised Southern California. I'm definitely a California beach boy, as you can tell by my music and the time I spend at the beach.

Mary: I can tell by your tan.

Steve: I definitely spend some time in the California Sun. As we are recording this, it's the end of March and we've only had two inches of rain since last July. So that tells you that we get lots and lots of sunshine.

Love the climate out here and love being close to Disneyland. Hey, I used to be a skipper on the Jungle Boat Cruise for a while. I like to think that I have a little bit of an entertainer's soul in me.

Mary: Do you have any Kids?

Steve: I have one child, Jeannie and she is my pride and joy and she will be 12 this September. My lovely wife Lori and I met (believe it or not) on a hiking outing with the Sierra Club.

Mary: So, you get out and do stuff. Where's woods in Southern California?

Steve: Ah well, what is interesting is that one of the largest city bound parks exist in Los Angles by the name of Griffith Park. There is an Observatory, Zoo, Museums and lots of hiking in the mountains. The Local chapter of the Sierra Club hosts hikes on weeknights in the spring with up to 500 people hiking in 20 or more groups. That's a lot of people!

Mary: Do you have a dog or any other pets?

Steve: No, not at this time. But I'm fond to Basset Hounds. They have such a sad face, with those long droopy ears and big, sad eyes. Their temperament is great, a kid could pull their tail and all they would do is turn with a sad, sorry-full look of "why are you doing that?"

*Mary Interviewing Steve*


*Z-89, Zaxxon for the CoCo 3*

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

Mary: Oh, you got to love them.

Now, how did you get start with computers?

Steve: Well, I got started back in the early 70's. Back in high school during my sophomore year I was taking electronics classes. I was not interested in learning about the old tubes and transistors; I wanted to know about the new digital stuff.

I started studying about the digital electronics and I was going a little faster than the instructor. When the instructor was teaching the class and was not sure how to interface something, he would then look in the back of the room to see if I was shaking my head yes or no.

Mary: They had digital electronics back then?

Steve: Yea, they had digital electronics back than, but no microprocessors. I built my very first computer back in 1974 using only discrete integrated circuits (I.C.) Its memory was 4 groups of three 7489 IC for a total of 64 (12-bit) bytes of memory and it could only do 8 instructions. (NOP, ADD, COM, AND, OR, SHIFTLEFT, SHIFTRIGHT, JUMP_IF_ZERO.) It was basically a fancy calculator that did not calculate very well.


*The first personal computer and the world would never be the same.*

About a month after I finished my computer I saw on the cover of January 1975 Popular Electronics magazine the Altair 8800, the first microcomputer kit using a real microprocessor. I believe that was my first "I could have had a V-8" moment in my young life. After spending almost two years building my "computer" (if you could call it that), there was a true computer kit for only $400.

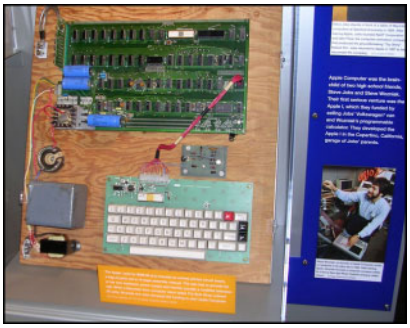
Mary: What changes did you see because of the Altair 8800?

New "personal" computers based on the Altair 8800 (and its S-100 bus) started showing up on the scene along with the Homebrew Computer Club up in Silicon Valley. From time to time, I would drive the almost 400 miles for the club meeting. Why would anyone drive so far for just a club meeting you may ask? It was there that I met people like Steve Wozniak (creator of the Apple computer) and Adam Osborne for the first time.

***"I built my very first computer back in 1974 using only discrete integrated circuits (I.C.)"***

I still remember Steve Wozniak bringing in his first Apple computer mounted on a wood board. While primitive by today's standard, it was like nothing anyone had seen before. You could just turn it on and start writing a BASIC program. It sounds a little a like your CoCo that you still play with today? You can thank Steve for that!

Later in 1975, I finished high school and moved on to College while working at Radio Shack store for the first time. At this time I was playing around with both S-100 bus computer based on the Altair-8800 and the SWPC computer based on the 6800, a new CPU from Motorola. I was moving from College to College looking for courses that'd aid me in new study of Digital Electronic, Micro-



*Steve Wozniak's Apple I computer*

Computers and software programming.

Mary: I was going to ask you about when you start programming. After all, you're mostly known for your games that you programmed.

Steve: Most people probably don't know that I'm primarily a computer hardware engineer. I love hardware! I spend just about as much time with a soldering iron in my hand as typing on a keyboard.

I started my studies in programming back in 1972, about the same time as digital hardware design. The high school offered classes on programming in BASIC and FORTRAN. FORTRAN used punch cards and was a real pain in the neck because the program had to be sent "Downtown" on punch cards. It would take days to get printout the program's run and to see if it worked at all. BASIC on the other hand was entered via a teletype (with a paper tape for storage) and you could run the program right away to see if it worked. You can tell which one I preferred.


*The shot fired in the Intel - Motorola CPU war, The SWPC 6800 computer*

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

It did not take long before I passed the level of what my high school could offer, so while at the age of 16 and still in high school I was off to College for courses on Advanced BASIC programming. I was also taking a Saturday (and sometimes after school) class offered by the school district for Assembly and Microcode programming. It was there that I learned the most important lesson of my professional life.

***"...it would take the knowledge of both hardware and software to become a success in computers."***

While at the district's computer lab, the MicroData mini-computer started having a few problems. Each day, MicroData would send out a technician to repair the computer. First was a hardware guy who found the hardware working fine, so it had to be a problem with the software. The next day, a software guy would look over the computer and find the software was up to date and loading great. Day after day, each of the techs gave the computer a clean bill of health till both showed up at the same time by accident.

As I watched them, I could see their combined knowledge was needed to deduce that the problem was a cross between hardware and software. What was going on was the ROM that held the Microcode was dropping just a few bits so the Microcode was not running right. Basically, a hard to detect bug in the hardware was making the software programs run very buggy.

*Steve Wozniak's second Computer, The Apple II*

They showed me that it would take the knowledge of both hardware and software to become a success in computers. Someday I must thank those two technicians for this life changing lesson.

Mary: It sounds like you did a lot with computer before the CoCo. What other personal computers did you write for before the CoCo?

Steve: Oh, there were a few computers here and there. Besides 8080 based computer like the Altair 8800, there were other computers based on the 6502. In early 1977 the Apple II and the Commodore Pet were introduced and I started playing with those. I still remember being at the 1977 West Computer Fair and seeing the introduction of the Apple II.


*The Apple II was followed by the PET*

In the summer of 1977, I took a little break from computers by working with a fellow magician at Nevada casino in a dueling magician magic show. One day before work, I stopped in the local Radio Shack store to get the new 1978 store catalog. It was like most other store catalogs except for a little insert on the last page for a computer call the Tandy - Radio Shack or TRS-80!

Radio Shack, the company that I work once before (at the time with over 5,000 outlets) was going to sell computers. Yes, a real personal computer, no kit, no loading programs in by switches. Plus the number place that you buy a computer has jumped by a factor of ten overnight. The personal computer was really here!

*TRS-80 Model one computer with expansion unit and hard drive.*

When my contract for the Magic show was over, it was back to Los Angeles and the start of my real career in computers. At this time I pulled a Bill Gates; dropped out of college. (More like I did not start my classes in the fall.) First, I started working for the Shack again but selling computer this time. I qiuckly became the top sales person in the district if not in the country because I knew how to write programs for my clients.

Mary: What types of programs did you write to help sell the TRS-80?

Steve: I did a couple of demos real fast but I also wrote application programs too. When someone would come in and ask "what could this computer could do?" I would then write the program for them to sell the computer.

One of the projects that I got paid for was a loan amortization program for a local car dealership. The program did such a great job for the loan manager that he started firing his staff. (The TRS-80 did a better job at filling out loan applications than people.). Back in the 70's, computers taking over jobs was a big deal and I hate to say that I was part of it.

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

It did not take long for me to learn that I could make more money if I spent all my time writing programs for clients. After about six months I moved on to my first paid programming gig at SofTape.

There I wrote programs for the Apple II and the TRS-80 including my first game, Space Ball.

Mary: Your first game, what did you write it in?

Steve: I should first correct myself and say that Space Ball was not the first game that I ever wrote. There were a few BASIC games that I wrote just for fun. Space Ball was my first game that was marketed and it was also my first assembly language game too.

Mary: The same company did Apple and TRS-80 stuff?

*A light Pen for a TRS-80*

Steve: Yes. Third party computer would and still do create projects for more than one type of computers. While At SoftTape I did work on both the Apple II and the TRS-80. Beside games, I also did drivers for Light Pens.

Mary: What is a light Pen?

Steve: In the years before the mouse (via the Mac) the only way to tell a computer what to do was via a keyboard and it was not a lot fun. So third party companies tried to market all types of new input device. The light pen was a small pen that you point to a spot on the screen and the computer would read where you were pointing at. But SoftTape's Bright pen was easy to use because of drivers that could be added with your programs.

The design of the Bright Pen's circuit was done by old time electrical engineer and very over done. It had a small pen with a phototransistor at the pen end and the other end was a pen holder with 7 transistors, 14 resistors and 10 other components. It looked nice but cost over $30 (in 1978 dollars) to make.

Because of my computer electronics background I understood just how the cassette input worked on the computer and to redesign the light pen to take advantaged of it. The new design used a Darlington phototransistor and resistor inside the small pen with a cable running to the battery and the cassette input jack. The design cost only $5.00 to build including labor.

***"While At SoftTape I did work on both the Apple II and the TRS-80 model I"***

Needless to say the designer of the original Bright Pen was not too pleased with my lower cost redesign because he was one of the founders of Softape. Never show up your boss they say, so I soon left the company.

Mary: So where did you go after Softape?

Steve: I have to give a lot of credit to my parents for my next move. They both grew up on small farms where you learned to depend on yourself. After all, if something broke on the farm, it was up to you fix it. Also, if you didn't know how to do something then you learned how.

They also ran their own business while I was growing up. It was a hard life working 6 days week for 8 to 10 hours a day. I would work there after school or on Saturdays doing what a boy could do to help out. It was there that learned what it took to run your own business.

After Softape, I follow the lead from my parents and start my first computer company, Computer Light & Sound. Softape let me keep the software that I wrote for them plus the redesigned light pen and I used them to start the company.

It was not long before every TRS-80 third party company that was selling a Light Pen based on my design and software. I knew that if I gave them my design and kept the software price low then I could corner the Light Pen Market. I can still remember walking the isles of the West Coast Computer Fair and seeing only my light pen at the third party TRS-80 booths and thinking that I had done well.

Besides the Light Pen, I also created three more video games, Galactic Fighter, Space Ball II and Galactic Fighter II. Space Ball was Breakout meets Space Invaders and Galactic Fighter was like Galaxian but predates it by 3 months.

One other Computer Light & Sound product that I'm proud of was SoftMusic. This was a two voice music driver that loaded with your BASIC program. It had two firsts, playing music on key through-out an 8 octave range. The second was play the music in two voice harmony and was unheard for a software only music system.

Mary: How long did you run Computer Light & Sound?

Steve: I ran the company for a very fun one and half years. The summer of 1980 was the start of a very colorful period in my career. But that's a story for the next issue of the CoConutz.

You can contact Steve Bjork via his website at coco.etechwds.com.


*A Space Ball for a TRS-80*

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER


# The real Donkey Kong on a CoCo? NO WAY!

**As many of you already know I went to the fest this year with a very special purpose. I was going to debut the newest and most amazing project by none other than the great Sockmaster!**

**Nick, John and I kept people waiting in the chatrooms on coco3.com. I was the main person to blame for all the torture amongst the others in chat. Two weeks before the release of John's secret project Nick and I would drop little hints.**

**There was a lot of discussion as to what John's latest project could be. Some thought it was a graphics editor others thought it was a game.**

**No one guessed that it was something so awesome. This game wasn't just a recreation but the actual game ported over to the CoCo3 with 512k ram. Most of you now want to know just how this all happened. Here is what John had to say.**

***had to have been several times faster and more powerful than the arcade game. ”***

Mary: What inspired you to do this?

John: I've been wanting to do something really neat on the CoCo for some years now, but I've never had the free time that I would need to do something really big. When some free time actually presented itself - I decided to use it! I hadn't really decided what I would do at first, except that I wanted to do something that had never been done on the CoCo before. That part is important to me. After thinking about it a bit, I finally settled on a game.

Mary: Why did you choose Donkey Kong instead of a different arcade game....

John: I had made a short list of about 5 arcade games that would be interesting to do on the CoCo. I didn't want to do something too easy. I didn't want to try something that turned out to be impossible after all. I didn't want to do a game that had already been done too many times on the CoCo. I wanted to pick a game that would look impressive on the CoCo.

After weighing all sorts of details Donkey Kong came out on top of the list.

Mary: What was your fist step in looking at how to do this?

John: First, find out everything you can about the original arcade game and hardware. See what it's capable of doing and then try to figure out how to get the CoCo to do the same thing.

Mary: You did this via an emulator right? What kind of emulator?

John: Not exactly. To do true emulation, the CoCo would have had to have been several times faster and more powerful than the arcade game. There are lots of emulators on the PC now specifically because it's several thousand times faster than these old games were. The CoCo on the other hand is from the same era as these arcade games. It *isn't* several times faster.The trick is to emulate only the parts of the hardware that is within the capacity of the CoCo to emulate, and the hardware must be emulated because it simply does not

1982 version of Donkey King by Chris Latham for a 512k CoCo3

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

exist on the CoCo.

The other parts that cannot be emulated... need to be converted into something that would work on the CoCo. The big part that can't be emulated is Donkey Kong's Z80 CPU & Z80 program code. The CoCo has a 6809 CPU, not a Z80. What I had to do there was reverse engineer the Z80 code and translate it into 6809 code. That way it can run on the CoCo, and still run the same way that it did originally.

Mary: Explain the reason for the colored lines while decompressing?

John: The decompression takes about 20 seconds to process. That's a long time to wait. If there was nothing happening on the screen, people might think the program was not working and reset the computer! The crazy colors are an easy way to show that the computer is actually doing something. Many demos on the Amiga and Atari ST used to use the same effect for the same reason. I figured it would be fitting for the CoCo to do it too.

Mary: Is this the identical game speed etc just ported over?

John: The game should play identically to the arcade. Same speed, same difficulty, same tricks and techniques.

Mary: Of course everyone wants to know what is up your sleeve for a next project?

John: Even I don't know the answer to that.


*First Level and it's dead on*


*Second Level of 1982 version of Donkey King by Chris Latham for a 32k CoCo.*


*Same level on 2007 version of Donkey King by John Kowalski for a 512k CoCo 3.*

Mary: Do you plan to release the emulator source code?


John: I'm not sure yet. If it looks like other people are actually in the process of doing similar arcade-to-CoCo conversions and if I decide this source code could be helpful to the CoCo community, I may. Everybody "just wanting it" and ganging up on me to ask isn't going to help, by the way...

Mary: Did you learn any secrets about DK along the way?

John: Lots! There's a short message to hackers in Donkey Kong's rom code. Did you know that Donkey Kong actually "runs out" of barrels when the time runs out on the first level?

I learned a number of gameplay tricks. You can actually pull out the rivets without crossing over to the other side - useful when you have the hammer and wouldn't be able to jump back. If you do it right, you can also jump off the top level of this stage and you'll just bounce back.

On the elevator level, you don't *have* to jump off the elevators - you can simply just walk off and land safely.

Mary: Do we have cheat codes?

John: There are no cheat codes in the game. I do have one POKE for the cheaters out there. Just add a POKE9665,230 before the EXEC in line 100 of the loader for unlimited lives. It kind of spoils the game, so I'd recommend not using it.

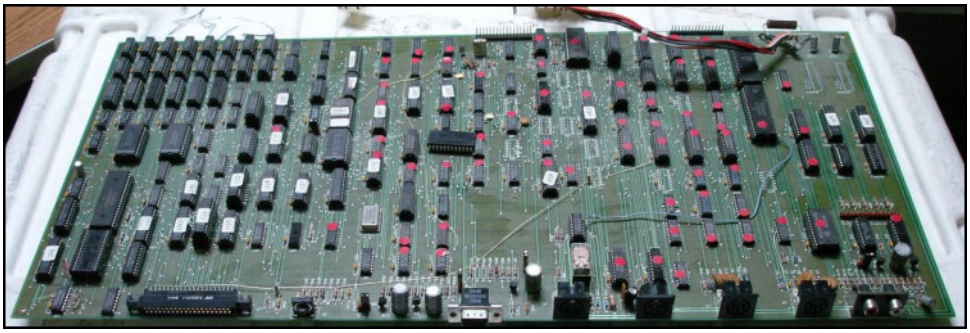
You can downlaod the game for free at John's website:

www.axess.com/twilight/sock


*Every level is dead on*

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

# The Color Computer 3 Prototype



**By Allen Huffman**

### Prologue - In the Beginning

On a warm August day in 1985, a Federal Express delivery truck pulled in to a parking lot in Clive, Iowa like it did almost every day. The driver retrieved a nondescript cardboard box from the back of the truck and carried it to the lobby. The box was signed for and left, then the driver returned to his route, unaware of the significance of what he had just been part of. The box, you see, had been sent by Tandy in Ft. Worth, Texas. The recipient was a small computer company called Microware Systems Corporation. The contents of the box were a secret prototype for a new computer which would be appearing the following year in Radio Shack stores nationwide: the Tandy Color Computer 3 (aka, the CoCo 3).

That was two decades ago - a lifetime in the computer world. Few specifics about what went on behind closed doors at Microware or in Tandy Towers are known. What is known, however, is that Microware had previously established a business relationship with Tandy to produce a version of their OS-9 operating system for the original Color Computer. This time, their involvement would go far beyond just doing another port of OS-9 to new hardware. It would involve them working on the onboard firmware to bring the new machine to life. Microware would be expanding Extended Color BASIC to take advantage of the new hardware.

But why Microware? In 1979, Microsoft (yes, that Microsoft) had done the original Color BASIC for the Color Computer so surely they would be the ones to continue doing so. But, by 1985, Microsoft had moved beyond being just a provider of BASIC and those types of projects just weren't compelling. Some speculate Microsoft would have done it, but it was just cheaper to have another company work on the project. In either case, Microware was likely chosen because they had previous experience working with Tandy and the CoCo on the OS-9 project. Since there were plans to bring out the next generation of OS-9 (Level 2) for the new machine, perhaps the economy of scale (a discount for doing multiple projects) did play a role in this decision. We may never know the full details, but regardless, in 1986 a new CoCo 3 began appearing at Radio Shack stores nationwide, and its new Extended Color BASIC featured enhancements done by Microware.

Although the story of how Microware had to patch and extend Microsoft's code is an interesting one, this is not that story. Instead, this is the story of the contents of that secret box. This is the story of the CoCo 3 that almost was.

### Part 1 - The Discovery

It was January 2005 and the large, three-story custom-built Microware building was finally being vacated by its original owner. Microware had ceased to exist as an independent entity in 2001 after it was acquired by Oregon based RadiSys Corporation. Over the years, the once thriving embedded operating system company had become a much smaller struggling company trying to compete in a market now filled with hundreds of competitors, including offerings from Microsoft and embedded versions of the free Linux. Although the building, completed in 1996, was once fully occupied by Microware staff, it had slowly been rented out as the company reduced in size. At some point, the building was sold and the former owner became a renting tenant. It was on this day that the last remaining Microware folks would be relocating to a much more appropriately sized rented office space a few miles away.

The move was somewhat emotional for those who had been with the company since the 1980s. Efforts were made to preserve any OS-9 related artifacts that might still prove useful, such as motherboards for any versions of OS-9 that were still supported. VME cards were salvaged and server racks were saved, but endless other pieces of ancient hardware were to be recycled. Large trash units had been brought in to the parking lot. Old PCs, SUN workstations, endless cables and old parts were being thrown in to them. A mountain of monitors was stacked high in the lobby, waiting to be picked up by the recycler. Decades of history had been rendered useless by the advances of technology.

One of the final areas to be cleared out was a small storage room in the basement known as "the morgue." Inside the morgue were some of the more interesting artifacts of Microware's past. Shelving units full of Compact Disc Interactive (CD-i) development systems stood across

from piles of old software disks and tapes. Endless VME I/O cards, motherboards and reference hardware sat under layers of dust next to boxes of blank EPROMS and serial cables. It was a place that, in the 1980s, would have been a hardware hacker's wet dream, but today it was just a room of ancient technology with no modern value or use to anyone.

Just like Noah and the Ark, two of each potentially useful item was to be saved. Anything that was no longer supported (or functioning) was to be sent to the great recycling center in the sky. Some historic items were allowed to be taken home, including an infamous Japanese video game system that ran OS-9 and featured mechanized 5 1/4" floppy drives and a fancy joystick. There were a few other pieces of unusual OS-9 hardware that escaped a crushing fate.

For instance, the CD-i machines also had some historic significance. Many were development systems used to create the tools Phillips and other companies used for making CD-i content. Others had been part of a shopping kiosk business known as Micromall, co owned by Microware in the 1990s. These CD-i machines were saved then sold off at the 2006 Chicago CoCoFest, hopefully helping them end up somewhere better than the recycle bin.

During all of this purging, a nondescript brown cardboard box was discovered. One of the remaining long time employees knew of its contents and made sure to set it aside. This box was the same box that Federal Express had delivered twenty years earlier. This box contained not one, but two Color Computer 3 prototypes and a few other surprises. The contents of this box have since helped us learn a bit more about what Tandy had intended the CoCo 3 be.

## Part 2 - From Prototype to Pre-Production

Before the discovery of the actual Color Computer 3 prototypes, the CoCo community had already seen what was being called "prototype CoCo 3s." A few years earlier, some pre-production CoCos were displayed at a CoCoFest convention. The CoCo Communityís official monk, Brother Jeremy, had acquired them somehow. They were the ones used by Microware for developing OS-9 and, we assumed, the Extended Color BASIC extensions. Externally they looked like the CoCo 3s we are all familiar with, but the motherboards inside were different. The GIME chips were earlier prototype versions, different from the ones found in later production units. Little else is known about these machines, but news of their existence spread through the community.

After hearing that "prototype" CoCo 3s had been shown publicly, one of the original Microware CoCo 3 developers made a comment that those couldn't possibly be the real prototypes because the real ones were still in storage at Microware. This was the first clue that there was something else still to be discovered. Something few had seen, and something hidden away somewhere in a box stored down in a basement.

When the box was opened, it was clear no one had seen or touched its contents for many years, and quite possibly not since 1986. The insides were dusty. The labels were faded and cracked. A small supply of bubble wrap was all that protected the contents. Inside were two large green circuit boards and three smaller ones.

The large boards were covered in chips and wires. The only thing that gave any clue that this was connected to the Color Computer was a series of familiar connectors on the back edge. The standard CoCo joystick, serial and cassette ports were there along with a cartridge connector. Elsewhere on the board could be found a keyboard connector, and further inspection of the chips revealed a few recognizable ones, like a 6809 processor. The amount of chips (on a board four or more times the size of a production CoCo motherboard) was staggering. The back side of the board was covered in dozens and dozens of long green jumper wires.

Two smaller CoCo cartridge boards were also found as well as an unidentified third board that didn't seem to plug in to anything. The cartridges matched one that had shown up a few years earlier at a CoCoFest that was thought to be some kind of Ethernet networking pak. The third mystery board carried a Copyright 1984 Tandy notice on it, indicating it was probably too early to be anything CoCo 3 specific. It was this set of five boards that was shown at the 2006 Chicago

CoCoFest, and this was when the next round of discoveries were made.

## Part 3 - Blue Sky CoCo

"Everything, even the CoCo, starts with a dream."

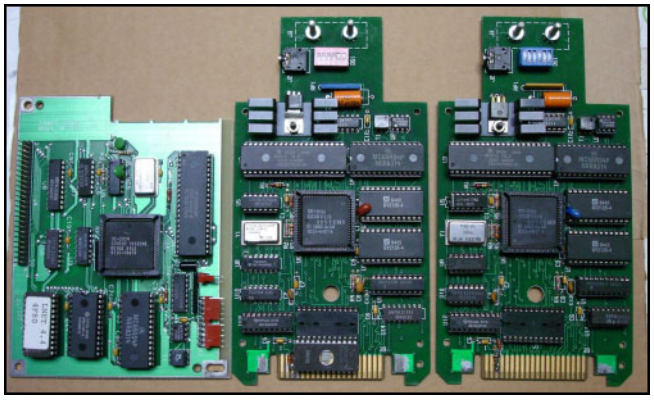
When Disney's Imagineers start designing a new ride or attraction for one of the theme parks, they initially

start with what they call the "blue sky" phase. That is, "the sky is the limit." Anything is possible, even if impossible. These initial concepts and ideas may be far grander than what is technically possibly, or perhaps possible but economically unfeasible. As the project continues, the budget (and often the realities of technology) whittles down the blue sky plans to something much more humble which hopefully will get approved and built. Disney fans know far too well how grand plans originally become much smaller realities, such as how Walt Disney World's Epcot "Space Pavilion" went from a full experience with a space shuttle launch to a space station, to just a simulator ride that didn't pretend to be anything grander.

This same approach is common in many areas of design, and likely played a role in the evolution of the Color Computer series. For instance, it is documented that a Deluxe Color Computer was planned but never released. Evidence of this includes references to a deluxe model in the Color Computer 2 manuals. Little is known about the features of this version other than a documented ability to enter BASIC commands in lowercase. Such capabilities never made it in to any official version of Radio Shack CoCo BASIC, but later models did include support for a lowercase display. There were also references to extra keyboard keys. Coincidentally, Radio Shack stores sold some keyboards as spare parts during this time. Theses keyboards had a few extra keys and could be plugged directly in to an existing CoCo. It is believed that these keyboards were designed for the never-produced Deluxe CoCo. Perhaps some day a prototype of this machine will surface.

It is possible that the CoCo 3 grew out of blue sky plans for the Deluxe CoCo, actually allowing more ambitious plans to be made than just minor improvements. All that known for sure today is that the Deluxe CoCo plans got far enough for keyboards to be manufactured and for manuals to be revised and printed.

To understand what was happening, it is helpful to look at what had already happened. Tandy has already evolved the original grey case CoCo several times. There were a few revisions to the original motherboard with the final versions supporting 64K without hardware hacks. A small run of white cased CoCo 1s was also produced which included an updated keyboard. Next was the CoCo 2 in a smaller white case with a similar keyboard,



though they were soon revised to have an improved keyboard which would continue to be used on all later models. There were numerous revisions to the CoCo 2 models, though the only significant feature was the addition of true lowercase for the "Tandy" branded units. (None of the models labeled as TRS-80s had this enhanced video chip.) There was also another minor revision that caused the need for Color BASIC 1.3, but the end result was a machine that was effectively no different than the original 1980 model other than in appearance.

To truly make a successor, Tandy needed something bolder than just a new keyboard and case. Game developers wanted to see enhanced graphics. Similar peer systems, such as the Commodore 64, had more colors and hardware sprite capabilities which allowed more advanced games to be created easier. Some of these capabilities were already available as expansion pak add-ons for the CoCo, but developers couldn't target those enhancements since the base CoCo did not have them. In order to be useful, the hardware would have to be integrated.

Looking at the lineup of add-on hardware paks sold by Radio Shack, certainly building in enhanced audio (like the Speech/Sound Pak) would be useful. The RS-232 pak would also have made a nice addition, effectively giving all those "power user" features to the base model. A "really deluxe" CoCo with better graphics would also need to support something other than an old-style television set. Other obvious enhancements would include more memory and speed.

Ultimately, the CoCo 3 that was released in 1996 only contained a handful of these blue sky items. Compromises had to be made to keep costs down. One of the original Tandy CoCo 3 developers, Steve Bjork, has stated that there was a requirement for the CoCo 3 to be produced at a lower cost than the CoCo 2 it was replacing. This ambitious economic goal certainly limited all the developer's requests for enhanced hardware.

When released, the production model CoCo 3 did contain better graphics (up to 640x480 with four colors, or 320x225 with 16 out of 64 colors). More memory was added, with the base model of 128K expandable to 512K. RGB-analog monitor output was added, as well as audio/video outputs for hooking to VCRs or composite monitors. The CoCo 3 could also run at double speed even in RAM mode, allowing a boost in performance for more than just BASIC ROM calls. (The

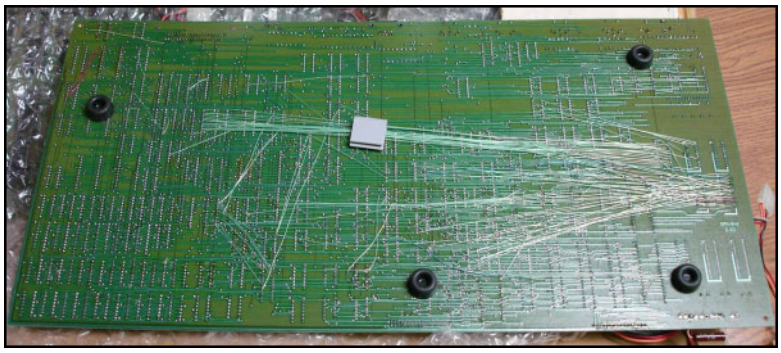
CoCo 1 and 2 had a "double speed" poke which sped up ROM access, but the so-called "triple speed" poke that sped up RAM access garbled the video display. The CoCo 3 allowed the "triple speed" poke to work for both RAM and ROM code without losing the display.)

Other additions were compromises. In lieu of real sound hardware, developers such a Steve Bjork lobbied for and received an enhanced IRQ timer. This didn't compete with the music chip in the Commodore 64s, but it did allow enhanced background sound effects using the existing CoCo sound capabilities. There was another IRQ that would have dramatically helped with RS-232 performance over the printer/ serial "bit banger" port. It is believed this was meant to substitute for a hardware RS-232 interface, but it was miswired so that potential was never realized. The list of other enhancements that would have been nice but didn't make it is something discussed to this day, usually under the guise of what would have been in a CoCo 4.

Overall, the CoCo 3 was a significant leap forward for the series. It contained better graphics, more memory, faster usable speed and even added the extra keys that would have been part of the Deluxe CoCo (but still no BASIC enhancements to allow entering commands in lowercase). The new monitor outputs and new color CM-8 monitor allowed using an 80 column screen, finally breaking away from the 1980-vintage 32-column display. It was a significant upgrade and one that developers took too quickly. The new generation of programs, from enhanced games with full color and background sound to the power of OS-9 Level 2, made the new model a more revolutionary a leap than from CoCo 1 to CoCo 2.

This brings us back to that box and the prototype within. By the time hardware is created, even if it's just a massive circuit board stuffed with chips and wiring, most blue sky goals have been eliminated. The goal of the initial prototype is to begin working on what will hopefully be produced later. Projects certainly continue to evolve, often based on feedback from working with the prototypes, but examining early designs can shed light on the intent of the designers at that moment in time.

As mentioned earlier, the CoCo 3 prototype contained the common ports found on all CoCos up to that point - cassette, joystick, printer, TV RF out, and cartridge. New RCA jacks were added for composite audio/ video



output, and a DB9 appeared for the new RGB-analog monitor. While the RCA jacks would make it to the production CoCo 3s, the DB9 connector did not. Instead, an odd square 10-pin header connector was added to the bottom of the machine. This was likely a cost reduction move since placing the connector there on the motherboard probably saved some layout money, and using a surface-mount header was cheaper than adding a DB9 port. Still, it does indicate that Tandy may have intended to use some kind of monitor that had a DB9 connector like other monitors of the day.

On a side note, the production CoCo 3 still contains one mystery related to the monitor port. Under the machine where the monitor plugs in is a square indention that could have fit some kind of small box. Perhaps there was an idea of converting the main RGB-A output to some other format via a converter box (maybe simplifying monitors between US and other parts of the world). Perhaps there was some other intended us that we may never learn about. Perhaps the CoCo 3 prototype will eventually give us a clue. (A more pressing mystery is why CoCo 3 software always asks Composite/TV or RGB? on startup, even though there was a way to detect if a CM-8 was plugged in.)

Something else learned by inspecting the prototype is that Tandy may have had much higher goals for their base model machine. The prototype has no place for RAM expansion. Instead, it is populated with 512K. This would have driven up cost and would have caused real problems during the RAM price crisis of the late 1980s when memory upgrades shot up by hundreds of dollars due to a fire at an overseas production facility. Looking back, releasing a cheaper 128K unit that could be upgraded later was probably a smart move though it ultimately led to few official Radio Shack products taking advantage of systems with that much memory.

Another interesting discovery was noticing a 1773 chip on the prototype. This chip was part of the CoCo floppy drive controller pak. The prototype had the floppy drive circuitry built in and contained a ribbon cable connector for the disk drive. Tandy must have wanted to integrate disk support in the base model, and perhaps had goals of shipping a floppy drive with the system or allowing the CoCo 3 to just plug in an external drive without needing the drive controller. Tandy actually did this very thing with their Tandy 1000 EX and HX PC compatibles. While those systems contained a built-in floppy

drive (5 1/4" and 3.5" respectively), there was also a port that allowed plugging up an external drive. More on this later.

Probably the most curious observation was that the prototype did not even have a GIME chip. The GIME was a custom IC created to handle things like graphics and memory. In the early prototype stages, before such an expensive chip could be made, designers created the functionality of the GIME using programmable PAL chips and other support hardware. It is unknown how much GIME support is on the prototype, but since Microware used it to create the Extended Color BASIC enhancements (which included the new graphics modes), and since the prototype had 512K, is believed to have implemented the graphics and memory controller that the GIME later would handle.

There could be other secrets in this prototype. Early developers, under Non Disclosure Agreements with Tandy, received pre-production CoCo 3s with pre-production GIME chips. There were two known revisions to the production GIME (86 and 87 revision), and the early development units are believed to have been just earlier (and buggier) versions of what was released. But, documents given to Microware during this project indicated that one of the specifications for the CoCo 3 was a 256-color mode. Steve Bjork has stated that this mode never existed in any manufactured CoCo 3s, and notes that the graphics hardware itself did not have enough bits available to represent a 256-color map. However, this early prototype may have had the basis for such a mode before it was deemed either too costly or, perhaps, too likely to compete with the Tandy 1000 graphics. The mystery of the specified 256-color mode may finally be unlocked in these early designs.

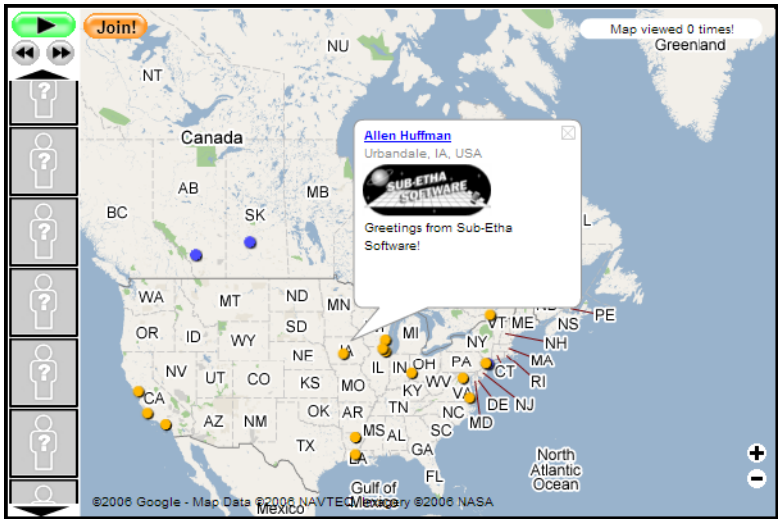
In a side note, the whole suspicion of a 256-color mode started when a former Tandy Color Computer product manager made reference to it years later. "Has anyone found the 256 color mode?" he asked. No one had, but noted Color Computer programmers John Kowalski

("SockMaster") and Australia's Nick Marentes were able to find abnormalities in the CoCo 3 schematics published by Radio Shack in the Color Computer 3 Technical Service Manual. As Steve Bjork has mentioned, there were not enough address lines for doing an 8-bit color, but the schematics showed some evidence of alterations in that area. There where two extra lines being routed away from the normal path. Perhaps there were plans to achieve the 256-color mode by some way that would allow accessing those extra lines? Nick was able to track down the original designer of the GIME, but he had no recollection of any such mode. It seems likely that this mode, if it ever existed, may not have even made it to the GIME stages.

So the prototype, while not quite a "blue sky" machine with enhanced sound and true RS-232 serial hardware, did certainly represent a somewhat nicer machine than what was actually released. Imagine a CoCo 3 with 512K standard, normal DB9 monitor port on the back, and a place to plug the disk drive ribbon cable in and still have the cartridge port available. This would have removed need for the MultiPak for the large number of CoCoists who used floppy and RS-232, or perhaps floppy and the Speech/Sound Pak.

When the prototypes are fully inspected and, hopefully, reverse engineered, there may be more mysteries discovered. Though the prototypes were supposedly working when they were packed away 20 years ago, until someone qualified has time to inspect them, no attempts are going to be made to power them up.

**The conclusion in the next issue of CoCoNuts!**




## Allen's CoCoFest Map

Drop by and place your mark on the map. This will not only give an idea on who all is going (or may be going), but it can be used to help coordinate ride sharing (if that helps some folks attend who might not otherwise get to attend).

http://www.frappr.com/cocofest2008

John (Sock Master) Kowalski
CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

## The 16th Last Annual Chicago CoCofest

My experiences there.

I have never attended a fest or anything like this. I was excited to go. I planned for several months and even kidnapped my younger brother to go with me. I had never even been to Chicago. My mother was upset I was going to Chicago, and my husband just couldn't understand why I wanted to travel so far to look at a bunch of old computers. I am very headstrong and told them both I was going weather they liked it or not. My mother demanded that I leave my son home with her and so I did. The drive up there was fairly nice until we hit Chicago.. Chicago was a huge surprise I was amazed to see all the big buildings and the sears tower crowded with , I think it was fog? I had never seen those above ground subway train things before. I thought that was neat that they had passenger trains running right next to the main roads. I did however miss a lot of the tolls cause I was unable to get into the right lanes. I figure I owe Chicago at least 5 dollars.

My intentions as to going to the fest were to debut Sockmasters Donkey Kong and video tape as much as I could of the fest. I was also extremely excited to meet and chat with Steve Bjork and talking him into an interview for this issue. I got there around noon and stayed until 5pm at the fest. In those short 5 hours I did hook up my coco3 for the debut of Sock's game.

When I first set it up people didn't know what to think. The first reaction was that it was a remake. Then people started to read the opening screen and someone said "holy cow this is the real Donkey Kong". Confusion set in as one of the guys started to play the game. So I let them soak it all in as I walked around the room and handed out copies of the game to everyone. I handed out a total of 30 copies. We had a little problem setting the game up at first. We tried to use Glenside's coco3 and couldn't get a joystick to work…. The whole time Allen Huffman is thinking to himself…..I knew it was an April fools joke…(Yes Allen I knew what you were thinking.) Finally I thought perhaps it was the Glenside's coco3's joystick port. I then went out to my car and got my coco3 and disk drive. I was right it was the joystick port on the other CoCo. So then we got the game going and the crowd gathered a little at a time. I think one of the kids played the game for almost an hour!

I was so pressed for time the only thing I really got to video tape was a scan of the room, and the seminar given by cloud nine. I will have that ported over by someone as soon as possible so others can see it. I found it very interesting how they explained everything. I also have on video Mark Marlette sitting behind his table with his monitor facing to the crowd for people to see as they walked by. The monitor had a Windows prompt on it! I thought that was rather amusing.

I also was asked to sit in on Steve Bjork's panel for his discussion of how the CoCo changed your life. During the discussion Steve brought out his Electric Crayon machine and Allen also showed his COCO prototype! It was real nice to see those items and hear the story that went along with it. Allen Huffman videotaped this event. If we bug him enough maybe one day all his fest footage will be published?

The last part of my day before I had to return to my family was when I sat down to talk with Steve. This interview was also videotaped by Allen Huffman and we used an audio copy to create a transcript for this issue. The funniest thing I have ever seen was watching Mark Marlette and the others belly crawl so they wouldn't disturb the recording.

I found everyone there at the fest to be wonderful and a pleasure to talk with.

I will be back next year but I will have to work out alternate transportation. My drive home was dangerous and very long. I will not drive to the fest again. I will however return next year.

Since I missed seeing one of the people I talk to the most online I asked someone to draw up a cartoon image of him. Diego Barizo shows the true spirit of wanting something so much he was willing to risk everything. We have all come together to help him pay his price for attending the fest. Diego has true spirit and brought a smile to all the faces at the fest. I wish I could have been there to give him a hug like I did some of the others. Diego I don't want to say you are honored because of your situation but I do want to say you are honored because of your determination. You took a 20 hour train ride to get to the fest. That alone is dedication and desire to the machines we all love so much. I know that we are all one big family and this was like a reunion for him and myself. That is why it is worth it to ride 20 hours on a train to get there.

So this picture is for you Diego. I will see you next year at the fest!

Mary Kramer

**Diego Barizo**

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

## The Asimov Awards - The final decision!

### A chat with Diego Barizo

Mary: How many people turned in a game?

Diego: Only 3 this year, and in one of the cases, I asked the participant if he wanted to include his already finished program in the contest.

I think another two were working on something, but I'm not that sure about that.

Mary: Did you extend the deadline? Why?

Diego: Yes, at the last minute, it was extended by some 2 extra weeks.

The official version is that some of the programs were almost ready, with the programmers working really hard to get them ready in time for the original deadline.

The fact that someone cared enough to work that hard just to be able to enter, was more than enough reason to do it. Also, at that time I was feeling a bit disappointed by the low number of entries, and I really wanted to have more than just 2 participants.

The unofficial, is that I was late checking the programs that were already sent, so I put those extra weeks to a good use.

Mary: Did you play each of these games?

Diego: Yes, more than a few times, and on the real deal (A CoCo, not an emulator). I had to learn a bit how to play the games, and try to make sure there were no obvious errors in them. If possible, I wanted to see all the screens and endings, but I must admit that being a very bad gamer, that was not something I expected to achieve.

Mary: What made you decide on the winner?

Diego: I think that what influenced me the most was the fact that it was the only game that took advantage of the CoCo 3 capabilities.

I tried to reach my decision based only on the "user experience". How big the files are, or how advanced the tech-


niques involved are not important at all.

The winning program seems to me to be a very polished product, easy to use and understand.

Most of the games (I think that all but one) are not original concepts. The only one that I haven't seen before is "Sentinel" That's why it deserved a special mention award.

Mary: Where there other judges?

Diego: No. Just me. I'm completely guilty and deserving target of any and all critics.

If I get a high number of programs for next year, I might ask for a helping hand.

Mary: What was the prize?

Diego: The original prize was $100 in cash to the winner. Eventually, a second prize of $20 was established.

And I still have to send the winners some certificate/diploma. Yes I am lazy.

Mary: How did you get the money for this?

Diego: Out of my pocket. The second prize came from a donation.

At the CoCoFest I traded diskettes with the programs for donations. I'm sorry that I wasn't able to be there on Saturday. My plans were to be there on a table and talk to people about the awards. But since I had already missed half of the fest, I just couldn't stay on a table. I also forgot to take the power supply for the laptop which I was going to use to show the games running in an emulator.

Anyway, after the word started to spread, people were actually offering me donations and asking for the diskettes.

Mary: Are there copies of the winning games?

Diego: Yes, I made a DSK file including all the programs that were submitted, plus another one that I would have submitted if someone else had organized the awards.

I also included a simple program that explains what this is all about, and works as a menu for the disk.

Mary: How can I get a copy?

Diego: You can download it from http://coco.sclaudia.net/AA.DSK

I still have to put a real link to it in the page http://coco.sclaudia.net/AA/aa.html

There is also an OpenOffice document that I used to make the disk sleeves at http://coco.sclaudia.net/disk_sleeve.odt

Mary: Are there copies of the other contenders games?

Diego: Yes, all the programs are there. It was always my intention to make all the programs available, even if I had to use a "flippie" or multiple disks.

Mary: Does the game work in an emulator?

Diego: yes all of them do. I use a little known one called "Bjork" (I believe it was also known as "Poco")

http://members.lycos.co.uk/poco6809/bjork/index.html

They should work OK in any other emulator

Mary: Does the winning game need a joystick

Diego: No, the only game that needs one is "Glove". All the other ones work with the keyboard.

Mary: What is the name of the winning game?

Diego: The winning program is "Mister Mind". A computer version of the classic board game "Master Mind."

Mary: Who wrote it?

Diego: Christian Bilodeau.

On a final note, 1 of the participants is from UK, and the other 2, from Canada.

Let's hope that Team USA shows up for the 2007 edition (Yes, this is a cheap shoot trying to make national pride another reason to participate.

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

# Asimov Award Winner Mister Mind 2007 By Chris Bilodeau


Mary:The sound is awesome, is that your voice?

Chris: Yes it is my voice. I recorded all with my PC microphone in a wave editor program and saved it as a wave. I used my Tandy CCR-81 to record the music sample.

Mary:How did you do that?

Chris: I recorded all the digital samples on the PC. First part was to record the song that would be played when the player wins the game. I was limited by the available memory as I didn't want a bad playback and I needed to fit the entire game into 128k (Asimov contest restriction).

It ended-up with around only 2 seconds of song samples looped into a 10 seconds song for a total of 5 to 6 seconds recording with the voice. Note that I had to place the song samples strategically in the coco memory to have it played like if it was one non stop song. After that, I recorded "Mister Mind" with added sound effects and finally all the game peg colors.

Once I got all that recorded as WAV format, I used a Coco program called Maxsound made by Gimmesoft in 1987 to convert the wave to a Coco file. After getting that Coco sound file done, I moved it back to the PC again to edit the samples memory location, to cut the unwanted blank space and moved it back again to the Coco. I've done those tasks couple times before getting the final wanted result. To play it, I used a small ML routine to call the sound from BASIC.

Mary: How did you come up with the idea?

Chris: I made some programs with digital sound back in 80's. I remember 2 of them, one was for the Coco Club annual year-end activity. It was a BASIC QUIZ, the computer was asking question then applauding when the right answer was selected. It was saying "BRAVO" at the end.

The other one was for Radio Shack, I connected the Coco to the infrared sensor placed at the store entrance. When customers were walking in, the computer was saying HELLO instead of the usual bell sound. It was funny, I remember some customers answering the computer thinking it was us talking.

Mary: Why did you choose mistermind?

Chris: I originally made that game in 1989 when I was still a kid. I liked to play that game back then but never completed it the way I wanted. The original version had a simple introduction page, no Digital sound, no Scoreboard, no instructions and the Gameboard was taking about 30 seconds to draw before each game. As well, the original version didn't have a game level 2. When level 2 is selected the computer can choose up to 3 times the same color instead of 4 different adding more challenge to the game. Last thing was the composite monitor colors (CMP). I had a CM-8 back then and only worked it out for RGB. Because of the poor image quality on composite screens and the details in the game, using equivalent CMP colors was making the game almost unplayable because we hardly could read the computer hints after each line (crosses). I had to select different colors as well as displaying the hints (crosses) in a different manner.

Mary: How long did it take?

Chris: I don't remember how many hours I worked on the original version and I'm not sure how much more for the actual version. I can say that I worked a lot, some non stop programming overnights not having any sleep (I usually prefer working full time on projects since it always takes time to get back into it when left for a little while).

Mary: How did you decide to participate in the Asimov contest?

Chris: I was reading Diego's website and found out about the Asimov contest last summer but didn't really think I would participate. I got back interest for the coco last July almost 13 years after I packed it up (took it out from time to time and played a bit with MESS around year 2000 but rarely). Couple months later, I went on the coco3 chat then met some more coconuts/friends as well as Diego. Somewhere in December I guess, after a chat session with Diego, I decided it was the time. I needed a reason to do so, and wanted to help the cause and maybe having a chance

to win. I worked it all in January and submitted the Version 2.0 in early Febuary.

After the winner was announced, I started working on the version 2.1 and got that one a week before the Chicago Fest giving Diego's time to update the game on the Asimov 2006 disks.

**Conclusion**

I had a great time working on that game and I would like to say congratulation to all other participants as I think they all did a fantastic job. Thanks to Diego for that contest initiative, I hope more people will participate next year. Have fun playing the games.


## CoCoNut introduction Getting to Know Jack Rodda

Ok. It's time to get back to programming on the 6809. I've been away from the CoCo world for about 23 years.

First, a bit of background. I was working for Falconbridge Nickel Mines Ltd. (a base metal mining company in Sudbury, Ontario, Canada) when I started out programming on an IBM 1800 process control computer in Sept 1969. My first 2 years were spent writing FORTRAN programs on this computer, and on a remote S/360. In July 1971, the IBM 1800 was replaced with a DEC PDP-8/E computer. Falconbridge personnel wrote the operating system and a FORTRAN compiler/interpreter for this new computer. I did mainly FORTRAN programming on this computer, but also taught myself Assembly language programming in my spare time, using a spare CPU that we kept as backup for the main computer that was controlling one of Falconbridge's ore-processing plants.

By 1974, I was doing mostly Assembly programming, and in 1976-77 did most of the coding for an executive for a dual PDP-8/A system being installed in our Nickel smelter. Between 1978 and 1990 I did a lot of Assembler coding for this system and another PDP-8/A, and did some enhancements to our in-house FORTRAN compiler/interpreter. During this time I also got some hardware experience helping our Instrumentation Dept. work on the interfaces for the PDP-8. I also got some exposure to microcomputers with an Apple II that was bought by our computer department and then 4 hardened industrial microcomputers based on the Commodore PET.

In 1980 or 81 I bought my first "personal computer", my CoCo1 with 4K of RAM and color Basic which I quickly upgraded to 16K and extended color Basic with Radio Shack parts and then modified to accept 64K RAMs. I still have this computer, although all the manuals and my EDTASM+ cartridge have been lost in the intervening years.

My extensive background in Assembly programming and DEC systems lead to an interest in Assembly language programming on the CoCo, so I bought an EDTASM+ cartridge and proceeded to start playing around with 6809 assembler, which I found to be very similar to PDP-11 assembler. In fact, it was the 6809 processor, which was called at that time "the 8-bit PDP-11", that attracted me to the CoCo in the first place.

One project that I toyed with for a while was a FORTRAN interpreter similar to the one that had been written for our PDP-8 systems at work. It can be most easily described as being similar to the P-code virtual machine used to implement UCSD Pascal on the Apple II in the mid 1970s, except our's was developed in 1970-71.

In the Falconbridge FORTRAN system, a compiler reads FORTRAN source statements and produces an intermediate code that a "virtual machine" or interpreter executes at runtime. Although the net result is not as fast as code which compiles to assembly code, the programs are usually very small- an important consideration when you only have 4K of memory, as we did on our first PDP-8s.

Falconbridge's first PDP-8 system had only 8K of core memory, a 256 Kword disk, paper tape reader/punch and an ASR-33 teletype as an operator's console. Our second system had two 256 Kword disks, and we thought it was marvelous. Subsequent PDP-8 systems were equipped with 1.6 Mword disks and 32Kwords of memory. All of our initial systems had a 4Kword executive, and allowed 4Kwords for programs to run in. Our later 32Kword systems kept the restriction of 4Kwords for all programs, using the additional memory for disk caching and keyboard and printer buffering as well as additional process control programs.

This enabled us to build multi-user, multitasking real-time systems on the PDP-8 that ran in 32K memory with disk buffering, 4 simultaneous non-process users as well as all the process control in the plant, and monitoring of environmental monitoring stations scattered around a 30 mile radius.

Benchmarks

Experience on multiple systems invariably leads to questions of "What'll she do?". Benchmark programs were becoming popular about this time, and we compared our "ancient" PDP-8 systems to the more modern Apple II, and the latest kid on the block, the IBM PC, and the even more impressive IBM PC AT, using a simple program that calculates all prime numbers between 1 and 1000. It's not sophisticated, but it translates easily into BASIC, FORTRAN, PASCAL and C, and gives a good idea of the relative performance of the arithmetic in the various systems.

Unfortunately, the CoCo with its interpreted Basic barely kept up to the IBM PC's interpreted Basic, and was no match for the compiled Basic on the PC, or even our PDP-8 FORTRAN.

Obviously, the next step was to write a CoCo version of our FORTRAN compiler/interpreter to see if I could pump up the performance. The easiest and fastest route was to write the arithmetic part of the interpreter (FORTRAN "virtual machine") first and just hand-compile the simple little program into interpreter code. As I recall, preliminary results were very good, with the 6809 easily matching the performance of the compiled Basic on the PC.

Unfortunately, that's as far as I went, and the work that I did has been lost in the intervening 20-odd years. I'm going to have to start over again, with what I can remember of the coding I did, and some borrowing of coding that I've done on the x86 along the same lines.

Unfortunately, along with my original coding, I've also lost my EDTASM+ cartridge and all my CoCo manuals, as well as the cassette recorder and all the cabling. The only thing I got back from my brother-in-law is the CoCo1 itself.

Once I figure out how to run these new-fangled cross-assemblers on my PC, I'll try to get started at this again, (although I don't know why- no one would want a FORTRAN system on the CoCo). Just something I'd like to do. I'll have to cobble up an RS-232 cable from the PC 9-pin to the DIN receptacle on the CoCo.

I've been retired for over 15 years now, having retired at 50 years of age on Jan 1, 1992. Since I retired, my wife and I have moved at least 6 times (4 of which were "double moves" - first into storage for periods ranging from 1 to 7 months, then into the second house). It's a wonder that I can find any of my old computer stuff at all.

This is starting to ramble, so I'd better close for now. Meanwhile, does anyone out there think there may be any interest in a simple FORTRAN IV-like language for the CoCo? It seems to me that all of the users on CoCo3.com are interested primarily in games, which certainly doesn't fit with FORTRAN.




# LITE PSYCLE

**by CaptCPU (captcpu@clubltdstudios.com)**

One of my favorite CoCo programmers of all time has got to be Richard Ramella. Each month when I was a wee lad, I'd eagerly open up the new issue of Hot CoCo and flip straight to Elmer's Arcade. Usually, I'd be punching in Mr. Ramella's latest game before even reading the latest goings on at Elmer's. The genius of his games came from the simplicity. Playing Dang It! and Broken Field Nightmare over and over contributed mightily to the nut case I am today.

Most of my programming endeavors at the moment focus around the same concept. I'm nowhere near as skilled or inventive as Mr. Ramella, of course. But I think a bit of visiting Elmer's each month rubbed off and the idea of creating a super simple, but immensely fun game that will run on just about any CoCo has a huge appeal. This isn't it, but it's a step.

This first offering was actually a test of Roger Taylor's Rainbow IDE for programming in Extended Color BASIC. Not exactly the use for which it was intended, but it certainly beats NotePad by leaps and bounds!

Lite Cycle is a variation of the game everyone probably writes up for the CoCo at some point. Kind of the "Hello, World!" program of games. In some forms it's called snakes or Tron-style light cycles, or what-have-you. But it makes an interesting beginner's programming exercise. Being a rank amateur, I figure it's a good place to start.

## Requirements

Lite Cycles uses smidgen more than

800 bytes (that's with comments, spacing, and formatting), so it should run on any CoCo, 4K and up, tape or disk. I don't think I used any ECB commands in there, so it should run on a Color BASIC machines as well.

I tested it on Ramona (CoCo 3, DECB 2.1, 128K) and Ol' Betty (CoCo 1, F-Board, DECB 1.1, 64K), as well as the CoCo 2B and CoCo 3 (NTSC) emulation in MESS.

Color Computer 3 users that have an RGB or VGA monitor will want to type:

PALETTE RGB

prior to running the game. Might even add a line there at the top:

1 WIDTH 32:PALETTE RGB

since the game will not run properly out of 40 or 80 column mode.

I also recommend the high speed POKE for those that prefer a little more action. (I play it exclusively at this speed. It's way more fun.)

CoCo 3: POKE 65497,0

CoCo 1/2: POKE 65495,0

Be sure to slow down before accessing the disk or tape:

CoCo 3: POKE 65496,0

CoCo 1/2: POKE 65494,0

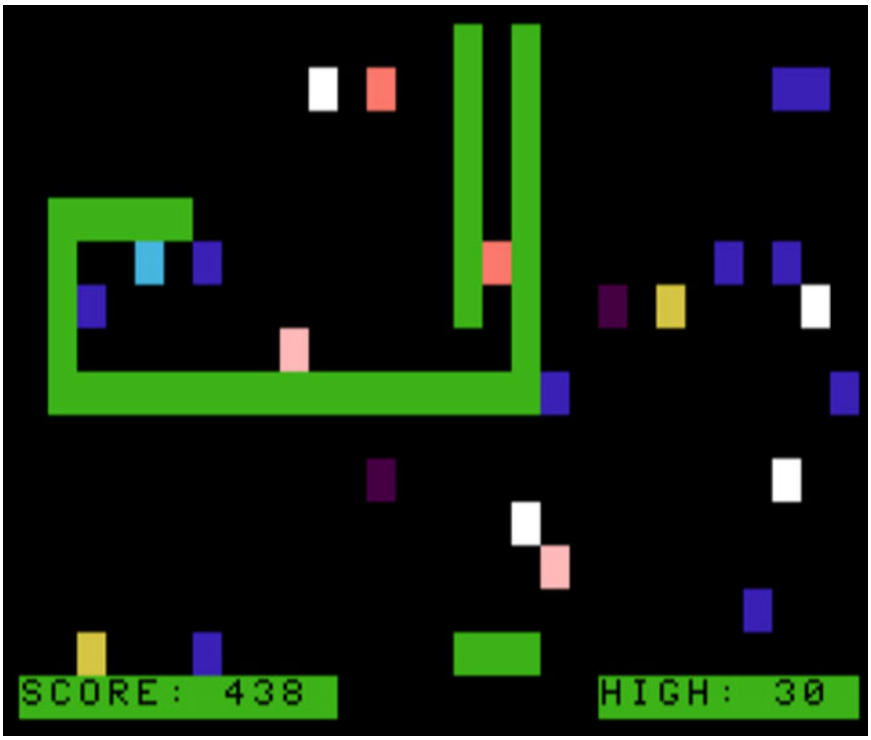
Some very old CoCo 1's may not be able to use this POKE.

## Game Play

Playing the game is pretty simple. You find yourself on the VidGrid ensconced in the ever popular and overly clichéd Lite Psycle. Your job is to drive about and snag all the colored squares you can before you bash into your own light trail. But beware, the evil video game controller gods have set upon the field nasty green squares that will smash you just as easily as smacking into your trail. Avoid them, drive like a maniac (using the arrow keys) and rack up as many points as you can!

I always liked weird scoring systems. So here's one! Every space on the screen your psycle moves you get 10 points. Easy enough. When you snag a colored square, though, you get 10 points plus the ASCII value of that square! Do you know your CoCo colors? If so, hitting those "higher" colors can net you more points.

In the spirit of simplicity, there's no exit. Hit BREAK when you're tired of it.




## The Program

The program itself is dirt simple and should be easily readable. This is probably more explanation than needed for this kind of program, but for completeness:

Lines 10 - 50 just set up a few things, like clearing the screen and setting X, Y, and score variables.

Lines 70-100 are kind of neat because they're "pushing" the CoCo's buttons, as it were. POKE 135 it like mashing a key. The values simulate pressing up, down, left and right keys. In this case, we're randomly setting the direction of initial travel for the psycle. (By the way, to get a more "random" number from the program, add the line 5 R=RND(-TIMER) to the listing. I didn't notice much difference, so I left it out, but for the purists, there ya go!).

Lines 110-140 just POKE in random dots around the screen. I chose 25 after some experimentation, but you can raise or lower this value according to your tastes. Line 120 looks like a mess, but it's pretty straight forward.

The value 1024 corresponds to the first (upper left) location of the CoCo's text (or low-res) screen. There are 32 times 16, or 512, locations (1024 through 1536) . So POKEing a value into that location makes the CoCo display the ASCII equivalent of the value to the screen. We want a random location somewhere within 32 columns (0-31) and 15 rows (0-14). There are 16 rows on the CoCo's screen, but we'll save the bottom bit for score information. The second part of the POKE selects what character is to be placed at the screen location selected previously. In this case we want a solid color block. Value 128 is a solid black block, to which we add

color using the formula found on page 292 of the Color Computer 3 Extended BASIC manual.

Lines 150-230 represent checking for a key press and then making sure the psycle stays on the screen. POKE 135 is a handy little guy for reading the keyboard (with some limitations, you can't read multiple key presses at the same time, for example). The values tested are for the arrow keys, up, down, left and right respectively (see pages 289 and 291 of the Color Computer 3 Extended BASIC manual). As a challenge, there's no border indicated and the psycle just wraps around the screen. So watch out!

Lines 240 and 250 check to see if a colored square has been hit. If it's not code 143, a green square, the player scores. If it is code 143, then the player has either hit a random green square, or hit their own light trail and the game ends.

Line 260 actually displays the next square for the light psycle's path, as calculated in lines 150-230.

Line 270 adds 10 to the score for a successful move, sans crash.

Line 280 prints the current score on the screen.

Line 290 finishes the main loop and sends the CoCo back for the next move.

Line 300 is necessary to clear out the keyboard, resetting it to its "no key pressed" state. Without it, the INKEY$ in the next line barfs and thinks a key is pressed. No waiting!

Line 310 waits while you lament your recent demise.

Line 320 checks to see if you scored enough for a high score and loads up that variable (to be displayed on restart way back on Line 20).

Line 330 is the do over line. Hit BREAK to quit.

**Play With It!**

One of the fun things about silly simple programs like this is modifying and expanding them, of course. Add joystick control, two players, have the psycle change colors when it hits a block (and then wreck it if hits the same color block next), or any number of bits and pieces.

In the end, this didn't turn out to be a good game, but it was an interesting programming exercise. I also got to play with Rainbow IDE and ended up with a game my daughter thinks is pretty neat. (Hey, she's 7.) It's been 17 years since I did this, so it's fun to get back into it and relearn. I hope others out there will do the same. And if you do, post your efforts!

Capt's CoCo Hut, http://coco.clubltdstudios.com - The CoCo Hut Blog http://cocohut.blogspot.com

```
----Program Listing ----
10 CLS0
20 PRINT@500,"HIGH:"HS;
30 'LIGHT CYCLE, CAPTAIN COMPUTER SOFTWARE, 2006
40 'HTTP://COCO.CLUBLTDSTUDIOS.COM
50 X=15:Y=7:S=0:B$=CHR$(128)
60 D=RND(3)
70 IF D=0 THEN POKE135,8
80 IF D=1 THEN POKE135,9
90 IF D=2 THEN POKE135,10
100 IF D=3 THEN POKE135,94
110 FOR L=1 TO 25
120 C=RND(8)
130 POKE (1024+RND(31)+(RND(14)*32)),128+16*(C-1)+15
140 NEXT L
150 P=PEEK(135)
160 IF P=94 THEN Y=Y-1
170 IF P=10 THEN Y=Y+1
180 IF P=8 THEN X=X-1
190 IF P=9 THEN X=X+1
200 IF Y<0 THEN Y=14
210 IF Y>14 THEN Y=0
220 IF X<0 THEN X=31
230 IF X>31 THEN X=0
240 IF PEEK (1024+X+(Y*32))>143 THEN S=S+PEEK(135):SOUND200,1
250 IF PEEK (1024+X+(Y*32))=143 THEN
PRINT@263,"crash"+B$+"crash"+B$+"crash";:GOTO 300
260 POKE 1024+X+(Y*32),143
270 S=S+10
280 PRINT@480,"SCORE:";S;
290 GOTO 150
300 POKE135,0
310 IF INKEY$="" THEN GOTO 310
320 IF S>HS THEN HS=S
330 GOTO 10
```

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

## How to read a disk directory in DISK EXTENDED COLOR BASIC

**by Bob Devries**

While reading the directory using a BASIC programme is not trivial, it is not difficult once you understand the structure of the disk directory track. Let me refresh your memories on what is where.

I'll start with a brief explanation of the structure of a Colour Computer disk. The disk is divided into 35 tracks (like rings) numbered from 0 to 34. Each of these tracks is divided into 18 sectors, which each contain 256 bytes (characters). This structure is repeated for each side of the disk. So the capacity of the disk is 35 * 18 * 256 = 161280 bytes per side. Take out the directory track, and you'll be left with 156672 bytes of storage. The directory track stores the file names and other pertinent information about the file.

The directory is stored on track 17 of the disk, starting at sector 3. On a standard 35 track disk, there are 9 sectors used for directory entries. The maximum number of files is limited not by the directory capacity, but by the granule table, which is stored on track 17 sector 2. This table has only 68 entries. If you're using a modified DISK EXTENDED COLOR BASIC that allows 40 track or 80 track disks, you will have comparatively more directory entries.

Each directory entry takes up 32 bytes (characters) of the directory track sectors, of which half is normally unused. The first 8 bytes are the filename, padded with spaces; the next 3 are for the extension. Next comes 1 byte for the filetype, either 0 for BASIC file, 1 for BASIC DATA file, 2 for BINARY file, and 3 for TEXT EDITOR file. After that is the ASCII flag byte. A flag byte is one that stores one of two values, like YES or NO, TRUE or FALSE, ON or OFF. It is 0 for normal binary saved files, and -1 (&HFF) for ASCII (text) files. Then comes a byte pointing to the first granule of the file (more on granules later), and then two bytes to show how many bytes in the last sector of the file (1 to 256). The next 16 bytes are usually zeros (&H00), but may be used for other information. Disk EDTASM uses that area for its own values, and some other DISK EXTENDED COLOR BASIC versions use it for the creation date and time. There may be other uses.

So, what is a GRANULE? It is sometimes known as a cluster, or chunk. In DISK EXTENDED COLOR BASIC, it is a block of 9 contiguous sectors, starting at either sector 1 of a track or sector 10. This is the smallest amount that can be allocated to a file, and amounts to 9 * 256 = 2304 bytes. The GAT (Granule Allocation Table) is a section of sector 2 of the directory track. It fills the first 68 bytes of that sector. Each byte is a granule, starting at 0 (track 0 sectors 1-9) and ending, on 35 track disks, at 68, which is track 34 sectors 10-18. The directory track is NOT allocated a granule, and is simply skipped, so entries in the GAT with values over 34 (&H22) actually point to one track more than the number would suggest.

So, how do we read a directory? The core of the programme is the use of the DISK EXTENDED COLOR BASIC command DSKI$. Its template is like this:

DSKI$ <drive>, <track>, <sector>, <string variable 1>, <string variable 2>

The first three variables are fairly self-explanatory. Drive number (0-3), track number (0 to 34) but I only use 17 for the directory track in my example, and sector number (1 to 18) but 3 to 11 for my example.

The string variables are there to store the data read in by the command from the disk sector. Why two? I'm glad you asked. Remember that the maximum length of a string variable in BASIC is 255 bytes. A sector is 256 bytes long! So 128 bytes are stored in the each string variable. For the purposes of my example programme, 127 bytes from the second string variable are concatenated to or added to the end of the first, and the last byte can be discarded, because it does not have any information in it that is needed to interpret the directory entries. Perhaps a more correct way would be to detect when the first string has been used, and simply copy the second string to the first. My version is however, simple, and it works!

So to read the first of the sectors containing the directory entries, the command will be:

DSKI$ 0, 17, 3, A$, B$

Of course, any string variable names could be used, and the drive could be any of the four possible drives (0-3).

There are 8 possible directory entries in each sector. I say possible, because not all the entries are always valid. DISK EXTENDED COLOR BASIC marks a file that it has deleted by setting the first character of the file name to 0 (&H00) as well as setting all the granules to free by making them 255 (&HFF). Also, when a disk is newly formatted, the directory track (and all the other tracks) is filled with 255 (&HFF), so when the end of the directory is reached, the next filename will have 255 as its first character. That means we can easily write code to skip deleted entries and stop when the end is reached, by examining the first byte of each filename extracted.

Now that I've covered the technical details of what is where, I can go on to the dissection of the programme which I wrote to display this.

First, some house-keeping and a title:

```
10 CLEAR 2000:REM ASSIGN ENOUGH VARIABLE SPACE

20 PRINT"DIRECTORY UTILITY"
```

Next, prompt the user for a drive number, giving the option to keep the default drive, which is found in memory at &H095A. Drive numbers

are only valid from 0 to 3. Variable DN$ is used to get the user response. This is converted to numeric in ND, bounds tested, and transferred to variable DN, to be used in later programme lines.

```
30 DN=PEEK(&H095A):REM DE-
FAULT DRIVE
```

```
40 PRINT"WHICH DRIVE
(ENTER=";DN;:INPUT")";DN$
```

```
50 IF DN$<>"" THEN
ND=VAL(DN$):IF ND<0 OR ND>3
THEN 40 ELSE DN=ND
```

Next, the option to output to the printer is given. The PR variable is set to -2 if printer output is wanted, or 0 if screen.

```
60 PRINT"OUTPUT TO PRINTER (Y
OR N)";:INPUT PR$
```

```
70 IF PR$="Y" THEN PR=-2 ELSE
PR=0
```

Next, a table header is printed.

```
80 PRINT #PR,"FILENAME EXT TYP
ASC 1ST LAST"
```

```
90 PRINT #PR,"                FLG GRN
SECT"
```

Here's where the real work starts. The track variable TK is set to 17, the directory track, and a FOR-NEXT loop is started, beginning at 3, and ending at 18. Line 120 does the actual reading of the data, as explained earlier. Line 130 concatenates 127 bytes from B$ onto the end of A$, giving a length of 255 bytes. For our purposes, the last byte can be discarded, as mentioned above.

```
100 TK=17:REM DIRECTORY AT
TRACK 17
```

```
110 FOR SE=3 TO 18:REM DIR
ENTRIES START AT SECTOR 3
```

```
120 DSKI$DN,TK,SE,A$,B$
```

```
130 A$=A$+LEFT$(B$,127)
```

Next, a FOR-NEXT loop is started to read the 8 directory entries in the sector that was just read.

```
140 FOR EN=0 TO 7:REM 8 DIR
ENTRIES PER SECTOR
```

Line 150 extracts the FILENAME from the sector. Note that for this example, the data is not stored, but this could be done by using an array. The offset into the sector is calculated from the entry number variable EN (0-7), multiplied by the number of bytes per directory entry, which is 32, plus 1, because string variables are indexed starting at 1. Filenames are always padded out to 8 characters using space characters (&H20). The MID$ command extracts 8 characters from A$ starting at EN*32+1, which for the first entry will result in a value of 1. Subsequent values will be 33, 65,…

```
150 FI$=MID$(A$,EN*32+1,8):REM
1ST 8 CHARS IS FILENAME
```

Similarly, the EXTENSION is extracted in line 160, except that this time, the offset has 8 added to it, and the number of characters extracted is only 3.

```
160 EX$=MID$(A$,EN*32+1+8,3):
REM NEXT 3 CHARS IS EXTENSION
```

Line 170 gets the FILETYPE of the file. Values can be 0 - 3 as explained above. Here, the ASC command is used to convert the string to its numeric equivalent.

```
170 TY=ASC(MID$(A$,EN*32+1+11,
1)):REM NEXT CHAR IS FILETYPE
```

Lines 180 and 190 do the same for the ASCII FLAG. This byte is usually displayed as either "A" or "B" when you type DIR, so I have maintained that convention here, using AF$ to receive the flag to be printed later.

```
180 AF=ASC(MID$(A$,EN*32+1+12,
1)):REM NEXT CHAR IS ASCII FLAG
```

```
190 IF AF=255 THEN AF$="A" ELSE
AF$="B"
```

Line 200 is used to find the first granule entry of the file in the GAT. Again, ASC is used to convert the string variable to its numeric equivalent.

```
200 GR=ASC(MID$(A$,EN*32+1+
13,1)) :REM NEXT CHAR IS FIRST
GRANULE OF FILE
```

Line 210 calculates the value of the number of bytes used in the last sector of the file. This is a two byte number, because the complete sector could be used, resulting in a value of 256 (&H0100), which is two bytes long. This is calculated by multiplying the first byte by 256, and then adding the second, to arrive at a number between 1 and 256.

```
210 LS=ASC(MID$(A$,EN*32+1+14,
1))* 256+ ASC(MID$ (A$,EN*32+1+
15,1)):REM NEXT 2 CHARS ARE
LAST SECTOR LENGTH
```

The next two lines, 220 and 230, are used to skip deleted files, and detect the end of the directory. Remember, deleted files have the first character of the filename set to 0 (&H00), and the end of the directory is reached when the first character of the filename is 255 (&HFF). If the file was deleted, we simply skip the output part of the programme, and continue the FOR-NEXT loop. If the end of directory is detected, we fool BASIC into ending the FOR-NEXT loop, by setting the counter variables SE and EN to one greater than their maximum count values, and then continuing the loop. This has the effect of quitting the loop.

```
220 IF LEFT$(FI$,1)=CHR$(0) THEN
240:REM SKIP IF FILE DELETED
```

```
230 IF LEFT$(FI$,1)=CHR$(255)
THEN SE=19:EN=8:GOTO 240:REM
SKIP IF END OF DIRECTORY
REACHED
```

Line 240 prints all the data we have collected either on the printer or the screen, depending on the value of variable PR.

```
240 PRINT #PR,FI$;".";EX$;" ";TY;"
";AF$;" ";GR;" ";LS
```

Following this we have two lines with NEXT statements, one for EN and one for SE.

```
250 NEXT EN
```

```
260 NEXT SE
```

The last lines, 270, 280 and 290 ask the user if another disk is to be done,

and either ends if no, or goes back to the beginning if yes.

270 PRINT"ANOTHER DISK";:INPUT YN$

280 IF YN$="Y" THEN CLS:GOTO 20

290 END

There you have it. Not as difficult as it first seems is it?

Here's a sample output from the programme:

FILENAME EXT TYP ASC 1ST LAST
FLG GRN SECT

GRN2TKSE.BAS 0 B 32 114

DIRUTIL .BAS 0 B 33 129

DIRUTIL .ASC 0 A 34 19

Next time, I will show how to find all the granules that are used by a programme, by searching through the GAT. That will also allow me to show the actual file size in bytes.

I hope this tutorial has shed some light on this subject for some people, especially those who are new to DISK EXTENDED COLOR BASIC.

# One-Liners by John Kowalski (Sock Master)

**This one-liner is “Snow in Spring”. Spring is basically here, but we had a snow storm over here a few days ago! Here is the CoCo version of it, with bonus spring-style snow.**

```
10 POKE65497,0:CLS0:DIMA(63),C(63),S(63):FORX=0TO63:A(X)=RND(28):C(X)=5:S(X)=1:
SET(X,31,1):NEXT: FORQ=0TO1STEP0:FORX=0TO63:A(X)=A(X)+S(X):SET(X,A(X)+1,C(X)):
RESET(X,A(X)): IFPOINT(X,A(X)+2)THENA(X)=0:C(X)=RND(8):S(X)=RND(0)/2+.5:NEXT :NEXTELSENEXT:NEXT
```

**This one-liner is "Mystify". It started out as a program that let you doodle on the screen, but it ended up evolving into a mystifying spiro-graph style doodler. Use the right joystick to "draw".**

```
10 POKE65497,0:PMODE4,1:PCLS:SCREEN1,1:DIMX(63),Y(63),C(63),U(63):FORA=0TO1STEP0:
FORE=0TO63:LINE(X(B),Y(B))-(C(B),U(B)),PRESET:X(B)=JOYSTK(0)*4:Y(B)=JOYSTK(1)*3:
C(B)=X(E):U(B)=Y(E):LINE(X(B),Y(B))-(C(B),U(B)),PSET:B=E:NEXT:NEXT
```

Both programs will work on either a CoCo 1,2 or 3. Just remove the POKE65497,0 or replace it with POKE65496,0 on CoCo 1/2.

## How I made my DSK file into a Real 5.25 CoCo Disk

**By Mary Kramer**

On the desktop of my computer I made a folder named "Mary" and then moved my DSK file named "COCO" and the "DSKINI" program into that folder. Then in windows I went to the START menu and clicked on RUN and typed in the word COMMAND and pressed enter.

Now that I am in DOS I typed CD.. Until I got a C: prompt. Then I typed in "CD MARY" to give me a c:Mary prompt. Then I typed DSKINI (spacebar)A:(spacebar)COCO.DSK. Should look like this "DSKINI A: COCO.DSK" A: is the drive letter of my 3.5 floppy disk drive. Coco is the name of my DSK file. Then I took the 3.5 disk I just made and put it in my Coco's 3.5 drive. Then I put a 5.25 disk into my Coco's 5.25 drive and typed in "DSKINI0" (ZERO is my drive letter on the coco) ( I have a dual 3.5 and 5.25 drive setup for my coco). now with both disks in their drives I typed "BACKUP0TO1" (that backup zero to one. one is my 3.5 floppy drive number)

Whallllaaa! I had a real working 5.25 floppy disk of my DSK file.

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

# The Coco & Music

## by Robert Gault

There have been a number of messages on Malted Media about using music in Coco games. Let's consider what is required to be able to do this. First, how can the Coco make sounds of any kind?

There are two pieces of hardware in all Coco models that permit generation of sounds, the Digital to Analog Converter (DAC) and the single bit sound output from the Peripheral Interface Device (PIA) at port $FF22. By choosing the correct PIA settings either of these "devices" can be routed to the audio connector (RCA jack) or cassette output. Having said this, what quality sound can be generated by these devices and is it worth listening to?

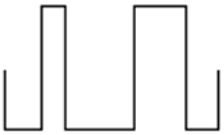
## What is Sound?

Sound is a rapid change of air pressure which vibrates the eardrum in the range of 20-20,000Hz (cycles per second). So, any variation in voltage in the above range sent to the audio outlet jack on the Coco will be perceived as sound. We can classify sound into two basic types, noise and music. The latter normally implies that there is a high degree of symmetry to the pulsations which tend to be sine wave like in character.

Above is a typical sine wave. Note that it is a smooth curve and will stay smooth no matter how much it is expanded. A musical signal of this type is said to have 0% distortion. As the curve becomes less smooth, distortion increases and large amounts make the listening experience very unpleasant. Can the Coco generate anything close to the above waveform?

## Coco Sound Output

The single bit sound sends either 0v or 5v DC to the sound amplifier. That's because a single bit at $FF22 is used to generate the sound. So the wave form must be a square wave of varying frequency but constant amplitude.



If the above waveform was symmetrical that is constant frequency, it would seem to be a tone with a very objectionable amount of harmonic distortion. The distortion in this case results from a square wave actually being a combination of many frequencies of sine waves that are overtones of the fundamental but that goes way beyond the scope of this article.

The above square wave is not symmetrical along the X axis so it is frequency modulated. A frequency modulated square wave can sound like a sequence of tones with large amounts of distortion. It is quite adequate for generating noise for games but is not desirable for music. Can the Coco do better than this?

The DAC used in the Coco is a six bit unit. Each bit represents a different voltage level and it is possible to represent 2^6 or 64 different levels. This means it is possible to create a digital version of the analog (smoothly continuous) sine wave.

As you can see, the wave is now a stair-step shape. If a complete cycle were drawn, there would be 32 steps above and 32 steps below the mid point on the Y axis. This clearly is not a true sine wave and it has considerable distortion but it is much better to listen to than a square wave of equivalent frequency generated by the single bit sound output.

## Evaluating DAC Quality

There is another method for evaluating the quality of sounds generated by the Coco or equipment such as Hi-Fi systems. Signal to noise ratio (SNR) is a much simpler technique easily calculated for the Coco while measuring distortion requires special test equipment. A good example of poor (low) SNR is a single voice at a party with many people talking loudly at the all at the same time. It is quite difficult to concentrate on that one single voice.

Since there are only two voltages from the single bit sound outlet, the signal to noise ratio becomes meaningless. Any random output has exactly the same volume as any desired output. There is also no possibility for a change in volume.

The DAC with 64 different voltage levels has an SNR of 1/64 or 1.56%. The normal method for expressing signal to noise is in decibels (dB) and the values are not linear but logarithmic. The formula for calculating SNR based on bits is SNR = 20olog10(2n) where n equals the number of bits. So the Coco DAC SNR is -36dB. To put this value into perspective, let's look at some common examples. A good Hi-Fi system has an SNR of better than -100dB. A properly recorded CD which uses 16-bit coding has an SNR of 96dB while a DVD with 24-bit coding and has an SNR of -144dB. A loud orchestra can reach 80-90 dB and amplified rock music 110dB so an SNR of a CD or DVD is very desirable for realistic music.

Since a change in volume of +6dB appears to sound twice as loud, it is a shame that the Coco did not use an 8-

bit DAC (SNR -48dB) as that would be perceived as twice as good a unit as the 6bit DAC. Eight bits is also the word size for the Coco so the use of an 8-bit DAC would make programming much easier.

## But How Does It Sound?

I doubt that many of us have our Coco audio routed to a Hi-Fi system so there will certainly be some loss caused by the cheap audio equipment in our monitors or TVs, and there is also the issue of frequency response. How fast can a Coco running at 1 or 2 MHz send data to the DAC and what will that translate into in terms of sound? We'll get to that shortly but first let's try to generate some actual sound.

Don't expect that the commands SOUND or PLAY will generate anything decent. If you look at the code for these routines, you will find that they send only three values to the DAC, low, medium, and high. That is not much better than single bit sound.

What we will do is write our own routines to generate 1-bit and 6bit sound of the same frequency and compare the results. The code is written to make the sound as clean as possible without regard to practicality in applications. The code as written, will compile with the Rainbow IDE but is easily altered for use with EDTASM. The code is intended for a real Coco not an emulator, but might still work with some of them.

Since you the reader may not have an assembler, the Basic programs can be used to create the binaries.

These programs will generate a square wave and 6 bit sine wave both about 500Hz. The single bit sound seems louder both because voltage at the RCA jack is about 1v peak to peak while the DAC signal is about 0.75v peak to peak. The significant amount of overtones in the square wave also contribute to its perceived loudness. For comparison, try the SOUND command which should be in-between single bit and 6bit DAC quality.

* Single Bit Sound - This does not work with the MESS v.111 but should with later versions.

```
        ORG    $6000
START   ORCC   #$50     turn off interrupts
        STA    $FFD9    fast cpu clock
        LDA    $FF23    PIA
        ANDA   #%11110111      program $FF22 as data direction
        STA    $FF23
        LDB    $FF22
        ORB    #2       program single bit sound as output
        STB    $FF22
        ORA    #4       return PIA to normal
        STA    $FF23
        LDA    $FF22    get current values
A@      LDX    #330     set timer
        ORA    #2       set bit high
        STA    $FFf22
B@      LEAX   -1,X     wait
        BNE    B@
        BRN    A@               balance loop
        LDX    #330     reset timer
        ANDA   #%11111101      set bit low
        STA    $FF22
C@      LEAX   -1,X     wait
        BNE    C@
        BRA    A@
        END    START
```

**BASIC Creator Program**

```
10 REM 1-bit SOUND
20 LI=80
30 FOR M=&H6000 TO &H6038 STEP10:SUM=0
40 FOR I=0TO9:READA$:VA=VAL("&H"+A$):SUM=SUM+VA:POKE
M+I,VA:NEXT:READ CHK:IFSUM<>CHK THEN PRINT"ERROR IN
LINE"LI:END
50 LI=LI+10:NEXT
60 SAVEM "1-bit",&H6000,&H6038,&H6000
70 END
80 DATA 1A, 50, B7, FF, D9, B6, FF, 23, 84, FB, 1616
90 DATA B7, FF, 23, F6, FF, 22, CA, 2 , F7, FF, 1714
100 DATA 22, 8A, 4 , B7, FF, 23, B6, FF, 22, 8E, 1262
110 DATA 1 , 4A, 8A, 2 , B7, FF, 22, 30, 1F, 26, 804
120 DATA FC, 21, F2, 8E, 1 , 4A, 84, FD, B7, FF, 1567
130 DATA 22, 30, 1F, 26, FC, 20, E4, 00, 00, 00, 663
```

BASIC Creator Program

```
10 REM 1-bit SOUND
20 LI=80
30 FOR M=&H6000 TO &H6038 STEP10:SUM=0
40 FOR I=0TO9:READA$:VA=VAL("&H"+A$):SUM=SUM+VA:POKE
M+I,VA:NEXT:READ CHK:IFSUM<>CHK THEN PRINT"ERROR IN
LINE"LI:END
50 LI=LI+10:NEXT
60 SAVEM "1-bit",&H6000,&H6038,&H6000
```

```
10 POKE &HFFD8,0:REM MAKES
SURE THE CLOCK IS 0.98MHz
20 SOUND 130,40
```

These programs nicely illustrate the problems of using the Coco to generate sound. The DAC program can't be made significantly faster without discarding or skipping data from the table and that would increase distortion. The single bit sound can be made 330 times faster which would increase the frequency well beyond our ability to hear it or even get it out of our speaker system.

So there is always a trade off between frequency response and distortion with the low sampling rate possible on the Coco. This works in the opposite direction as well. We are not going to get very far creating music from a sine wave table even though Dennis Kitsz did an amazing job with his 4 part harmony music editor, Quaver, published in The Color Computer Magazine. We will probably want to capture snippets of sound with a cheap microphone and store the sound files on disk for later use. The DAC can be used to measure voltage (that's how the joysticks work) and a waveform can come from a tape recorder, radio, or microphone via a preamp. The problem is that the fastest ml programs (properly balanced so they don't introduce errors) can't read the DAC (in this case ADC, Analog to Digital Converter) faster than that required for about 8-10 kHz tones. In any case, the amount of data generated would rapidly flood the Coco memory even with add-on boards of 2-8Megs. Storage of the data would require a hard drive system.

There was one fairly good commercial sound editing system for the Coco3, "Studio Works" by Jeff Noyle sold by Oblique Triad. It took the input of the ADC (6-bits) and stored short snippets of sound in memory. These could be saved to disk and later used by Basic or by machine language programs.

```
70 END
80 DATA 1A, 50, B7, FF, D9, B6, FF, 23, 84, FB, 1616
90 DATA B7, FF, 23, F6, FF, 22, CA, 2 , F7, FF, 1714
100 DATA 22, 8A, 4 , B7, FF, 23, B6, FF, 22, 8E, 1262
110 DATA 1 , 4A, 8A, 2 , B7, FF, 22, 30, 1F, 26, 804
120 DATA FC, 21, F2, 8E, 1 , 4A, 84, FD, B7, FF, 1567
130 DATA 22, 30, 1F, 26, FC, 20, E4, 00, 00, 00, 663

* DAC generated sine wave - works with MESS
        ORG   $6000
START LBRA  BEGIN
*SINE WAVE DAC values

DATA    FCB   127,131,135,139,143,151,155,159,163,167
        FCB   171,175,179,183,187,191,195,199,203,207
        FCB   211,211,215,219,223,227,227,231,235,235
        FCB   239,239,243,243,247,247,247,251,251,251
        FCB   255,255,255,255,255,255,255,255,255,255
        FCB   255,251,251,251,247,247,247,243,243,239
        FCB   239,235,235,231,227,227,223,219,215,211
        FCB   211,207,203,199,195,191,187,183,179,175
        FCB   171,167,163,159,155,151,143,139,135,131
        FCB   127,123,119,115,111,103,099,095,091,087
        FCB   083,079,075,071,067,063,059,055,051,047
        FCB   043,043,039,035,031,027,027,023,019,019
        FCB   015,015,011,011,007,007,007,003,003,003
        FCB   000,000,000,000,000,000,000,000,000,000
        FCB   000,003,003,003,007,007,007,011,011,015
        FCB   015,019,019,023,027,027,031,035,039,043
        FCB   043,047,051,055,059,063,067,071,075,079
        FCB   083,087,091,095,099,103,111,115,119,123
        FCB   127

BEGIN   JSR   $A976   DAC AUDIO ON
        ORCC  #$50    TURN OFF INTERRUPTS
        STA   $FFD9   FAST CPU CLOCK
        LDA   $FF20
        ANDA  #3      KEEP ONLY CASSETTE AND RS-232 BITS
        PSHS  A       SAVE DATA
        LDY   #$FF20  POINT TO DAC PORT

A@      LEAX  DATA,PCR      POINT TO SINEWAVE TABLE, 5cycles
        LDB   #BEGIN-DATA   GET # OF ENTRIES              2cycles

B@      LDA   ,X+     GET DATA BYTE
        ORA   ,S      ADD CASSETTE AND RS-232 BITS
        STA   ,Y      SEND TO DAC
        DECB  UPDATE COUNTER
        BEQ   A@
        NOP           BALANCE THE LOOP FOR LEAX & LDB
        NOP           7 cycles
        BRN   B@
        BRA   B@
        END   START
```

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

## Hardware Hacking

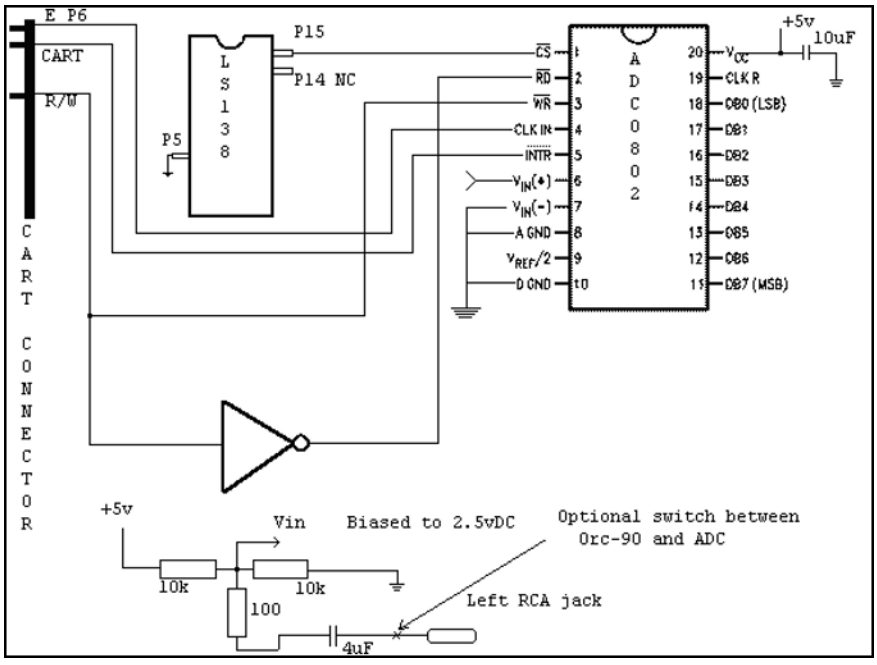
The only way to get around the hardware limitations of the Coco is to add new hardware. Tandy did this with the Speech/Sound and Orchestra-90 cartridges. They take over the task of sound generation from the CPU and DAC. The Orc-90 pack contains the equivalent of two 8-bit DACs (actually just resistor arrays on chips) and produces both stereo and higher quality sound. The Orc-90 pack also provides a perfect platform for adding a new ADC to the Coco to simplify and increase the sampling rate for digitally recording programs.

I chose the ADC0802 which used to be sold by Tandy (276-1792). This chip (except for speed) was ideal as it could be directly connected to the 6809 data bus. Since I needed to both read and write to the chip, I had to add read addressing to the Orc-90. I chose to piggy back another 74LS138 on top IC3 of the Orc-90. A low profile IC socket was soldered to IC3 except for pins 5, 14, and 15 which had their legs bent up. Pin 5 was used as an "enable" line and grounded, pin 14 was left unconnected, and pin 15 was connected to pin 1 (CS) of the ADC.

One additional chip was required for the new circuit, some type of inverter. The ADC required separate read write lines having the same logic values. The 6809 systems use a single read/write line giving read and write opposite logic values. The circuit below shows a TTL inverter but a 74LS02 (NOR) with one input grounded was actually used, the other connected to the pak R/W line (P18).

The data lines D0-D7 of the ADC were connected to the Orc-90 ROM (IC1) pins 11-19 using ribbon cable as a simple way to access the 6809 bus. The ADC clock was connected to the pak E clock line (P6) and the pak CART line was connected to the ADC interrupt line (P8). A simple resistor divider with an isolating capacitor was used to create a line level input for an AC signal and bias it to 2.5 volts.

```
BASIC Creator Program
10 REM DAC
20 LI=80
30 FOR M=&H6000 TO &H60DF STEP10:SUM=0
40 FOR I=0TO9:READA$:VA=VAL("&H"+A$):SUM=SUM+VA:POKE
M+I,VA:NEXT:READ CHK:IFSUM<>CHK THEN PRINT"ERROR IN
LINE"LI:END
50 LI=LI+10:NEXT
60 SAVEM "DAC",&H6000,&H60DF,&H6000
70 END
80 DATA 16, 0 , B5, 7F, 83, 87, 8B, 8F, 97, 9B, 1184
90 DATA 9F, A3, A7, AB, AF, B3, B7, BB, BF, C3, 1770
100 DATA C7, CB, CF, D3, D3, D7, DB, DF, E3, E3, 2142
110 DATA E7, EB, EB, EF, EF, F3, F3, F7, F7, F7, 2406
120 DATA FB, FB, FB, FF, FF, FF, FF, FF, FF, FF, 2538
130 DATA FF, FF, FF, FF, FB, FB, FB, F7, F7, F7, 2514
140 DATA F3, F3, EF, EF, EB, EB, E7, E3, E3, DF, 2342
150 DATA DB, D7, D3, D3, CF, CB, C7, C3, BF, BB, 2038
160 DATA B7, B3, AF, AB, A7, A3, 9F, 9B, 97, 8F, 1646
170 DATA 8B, 87, 83, 7F, 7B, 77, 73, 6F, 67, 63, 1202
180 DATA 5F, 5B, 57, 53, 4F, 4B, 47, 43, 3F, 3B, 770
190 DATA 37, 33, 2F, 2B, 2B, 27, 23, 1F, 1B, 1B, 398
200 DATA 17, 13, 13, F , F , B , B , 7 , 7 , 7 , 134
210 DATA 3 , 3 , 3 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 9
220 DATA 0 , 0 , 0 , 0 , 3 , 3 , 3 , 7 , 7 , 7 , 30
230 DATA B , B , F , F , 13, 13, 17, 1B, 1B, 1F, 198
240 DATA 23, 27, 2B, 2B, 2F, 33, 37, 3B, 3F, 43, 502
250 DATA 47, 4B, 4F, 53, 57, 5B, 5F, 63, 67, 6F, 894
260 DATA 73, 77, 7B, 7F, BD, A9, 76, 1A, 50, B7, 1249
270 DATA FF, D9, B6, FF, 20, 84, 3 , 34, 2 , 10, 1146
280 DATA 8E, FF, 20, 30, 8D, FF, 34, C6, B5, A6, 1470
290 DATA 80, AA, E4, A7, A4, 5A, 27, F1, 12, 12, 1263
300 DATA 21, F3, 20, F1, 00, 00, 00, 00, 00, 00, 549
```



All parts except for the new 74LS138 were mounted on copper backed perf board and easily fit into the Orc-90 case. In my case, I just sacrificed the Orc-90 left channel but one could easily use a toggle switch to select either the Orc-90 left output or the ADC input.

### How Does It Work?

The above circuit worked but not as well as I had wanted. The problem was the low conversion speed of the ADC. I'm sure that those of you that like hardware projects could find a faster unit.

The chip is rated at a clock speed of 640 kHz which makes the optimal conversion rate 8767 measurements per second. If nothing were done about this, the maximum frequency without aliasing distortion would be 4000Hz resulting in terrible audio.

I connected the ADC clock to the Coco E clock line resulting in a rate of 1.79MHz. The maximum rated clock of the ADC is 1.46MHz but the chip worked anyway. This faster clock gave a conversion rate of 24kHz which meant alias free signals up to 12kHz or AM radio quality sound.

The drawback of the faster clock was a loss in accuracy of conversion. The circuit should give a reading of $80 with no input (2.5v bias) but read $87, a 5% error. At a clock of 0.89MHz, the circuit read $80 with no input, a 0% error. I have not determined if the error is constant, linear, or other.

In actual use, the circuit sounds like a good portable AM/FM radio with software that reads the ADC and sends the data to the Orc-90 DAC. If you send requests to the editor of this publication and there is enough demand, I'll publish a program that demonstrates how to use this hardware. For now, all that is necessary to know that the ADC trigger is at $FF7A and the data port is at $FF7B. That means you tell the ADC to get a sample with a write of anything to $FF7A and read the 8-bit result at $FF7B. With the ADC in use, you can still send data to the Orc-90 DAC right channel by writing to $FF7B.

Put an optional switch as indicated in the schematic, and you can select either the ADC input or Orc-90 left channel output. Another option would be to replace the passive input with an op-amp having a volume control so that weak signals can be amplified. As designed above, a full input signal needs to be 5v peak to peak which is somewhat larger than a typical line output. A Hi-Fi preamp should work but a direct feed from a CDROM drive or sound card may be too weak.

### What Have We Gained?

The addition of an 8-bit ADC to the Coco unfortunately does not give you a bed of roses. You can record much better sound but are severely restricted by the Coco storage capabilities. Maximum length sound clips are about 21 seconds and require 442,368 bytes of storage. That's equal to 96 tracks or 2 1/2 forty track disks. In short, we are not much further ahead than we were using SOUND or PLAY.

Hope you had fun with a project that can teach us much about a very interesting topic and open doors to new ones. With this new hardware, I have written a program to turn the Coco into an oscilloscope permitting measurement and display of signals with frequencies under about 650Hz. If you are willing to sacrifice sound quality (significant drop in sampling rate), you can use Continuously Variable Slope Delta Modulation (CVSDM) to compress eight bits into one bit in real time. Using CVSDM, I have written a program that extends recording capacity to just over three minutes although the sound is not very good.

Want to read about these programs and see their code, contact the editor for more articles.


## What's happing at CoCo3.com, The Color Computer Super Site!

### What is the V.R. (Virtual Reality) CoCo 4 Project?

For years, many of us in the CoCo community have talked about what we would love to see in the next version of the Color Computer. But the sad truth is that Tandy stopped the production of the CoCo almost 20 years ago so there is no hope of a true CoCo 4.

In the past two decades we've seen PC and Mac system become so advanced that they can emulate a CoCo 3 system with plenty of power left over. Why not use the extra power to emulate a more powerful CoCo? That is the cornerstone of the V.R. CoCo 4 Project.

This is not a hardware project! The goal of the project is to create the next level of a CoCo by using software only via an emulator. Some of the features of the V.R. CoCo 4 will be faster speed, better graphics and sound.

What are your ideas on the project?

Add your input in the V.R. CoCo 4 section of the forms at coco3.com.

CoCoNuts! May 2007 Volume 3 Issue 2
THE LIFE AND TIMES OF THE COLOR COMPUTER

## Donkey Kong for the Coco3
## By John Kowalski

**Game Review By Brian (Briza) Palmer .**

We die-hard game players have waited 20 years for a version of Donkey Kong to call our own. Sure we did get a clone of "Return of Junior's Revenge". But it was never classed as a classic in the Coco gamer's field. What we needed was the Classic of all Donkey Kong games to be done for the Coco3. And only the master of the impossible, SockMaster, would dream of doing such a port.

But what a way to do it? Not do a clone, but an actual port of the game using the Arcade machines Z80 code translated to 6809. And doing hardware emulation into software based code then routed to the Coco3 hardware that could do it. I'm not a Techy by nature but that's the best way I can describe the achievement Sock's did in doing DK.

Now onto what Donkey Kong is and what it means on the Coco3. All I can say is: can any other 8-bit platform do Donkey Kong better then the Coco3 version? Maybe they could. But I think that it is only a dream on those machines.

The aim of this game is to rescue your girlfriend from the clutches of Donkey Kong. Who it seems was your former pet in a cage until he escaped. So to get back at his former owner (Mario) Donkey Kong decides to kidnap Mario's girlfriend Alice (I think this is her real name in the Japanese game Market).

This game has all the original screens. From the intro screen to the intermission screens and game screens.

First game screen you see is DK carrying Alice up the long ladder to the girders screen and this is where the fun and excitement begins. You start at the bottom left of the screen in which you have to jump, bash and run your way to the top girder. And just when you think you have rescued Alice DK picks her up and runs to the next stage of this game.

Some hints for getting to the top faster. If you time your run off from the left of the screen, you can actually just run & jump to get to the top no need to use a hammer at all. Here's another nifty feature which only the original did: you can actually run behind the burning oil barrel at the start of this level. But once its alight, you can't jump past it, only run. In this stage you have to avoid the barrels that DK throws at you and the flames that chase you after a barrel hits the oil drum.

Second stage involves you collecting the treasures (Alice's Belongings), and collecting the rivets to get to Alice. I found this to be the easiest stage to finish. No hints for this stage. Here you have to avoid the flames.

3rd Stage (Elevators), now this I would have to say is the hardest for me to beat; I always seem to get a hourglass landing on top of my head. Object is to collect the treasures; umbrella and hat and just make it to the top girder where Alice is calling out to you. Here you have to avoid being hit by an hourglass and flames.

4th Stage (Conveyors). This I would rate as the second hardest to beat, and this is the final stage in this game also. You never actually get to rescue Alice, as you always start at the beginning again. It just gets harder to beat each stage.

Here you just have to collect the treasures if you want, these being an umbrella, a hat, Alice's purse, and some nice tasty pies, then jumping over the flames and avoiding DK on the top conveyor.

By the way, this DK game has the US and Japanese versions. I find the US is harder than the Japanese versions.

Controls are right joystick to run and jump, and directional movements.

You also have the option to start with 3 men standard or increase your men to 6. 2 monitor displays are available, RGB or CMP.

After you select your options press the right joystick button to start. Best part is the musical tunes. You feel like you're beginning the game on an original arcade machine.

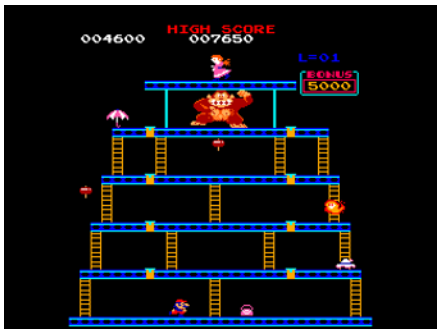
Here are the Specs for DK:

Graphics screen: 256x225 (Horizontal) x16 colors
6 Bit DAC for sound.
512k ram is needed.
And by what Sock's said, all game data (800KB) compressed down to fit 1x157kb single sided floppy diskette is just amazing.

Sorry Sock's but I pinched your screenshots of DK. My screenshots just can't do this game justice. Have to use the best for this game review.

I hope you enjoy this game review. If you haven't tried DK yet then what are you waiting for? Make a real copy and crank up the coco 3 with the volume turned up full. Sit back and enjoy the trip back in time.

Until next time, happy CoCoing.

## Cloud-9's DriveWire

**For the TRS-80 and Tandy Color Computer**

**Review by CaptCPU (captcpu@clubltdstudios.com)**

If you've ever wanted to add mass storage to your Color Computer, but didn't want to enjoy the expense of adding an IDE or SCSI interface, the super geniuses at Cloud-9 have just the thing. Created by Boisy Pitre, it's called DriveWire and it turns your PC into a big fat hard drive for your Color Computer. Once installed, storing and retrieving just about everything you've accumulated for your CoCo over the last three decades is just a few commands away.

The DriveWire package includes both PC and CoCo software on disk, an instruction manual, and a serial cable for connecting the two machines together. The PC software comes on a 3 1/2" disk and includes the server software. The CoCo software comes on a 5 1/4" disk. It includes a special version of HDB-DOS for Disk BASIC users as well as NitrOS-9 loaders. The cable included has a PC serial plug on one end and the round (RS-232) CoCo serial plug on the other. The user manual included is nine single sided pages and details installation and operation.

**Installing DriveWire**

There are no hardware modifications required to use DriveWire. Your CoCo will need its serial port free and a 5 1/4" disk drive. I tested the system with a stock CoCo 3 at 128K and dual FD-501 disk drive/controller plugged directly into the pak slot. I also ran tests using a 512K CoCo 3 with home-built 40-track 5 1/4" drives on a FD-501 controller plugged into a Multipak Interface. There was no noticeable performance difference between the two CoCos using DriveWire. The system will also work on a Color Computer 1 or 2, though at a slower baud rate. CoCo 1/2 operation was not tested for this review. (I hooked it, was able to transfer a file from the PC, but didn't have time to do anything more thorough.)

Your PC will need a 1.44MB 3 1/2" drive and a serial port installed. The PC I tested on was an old Pentium 4 1.44 GHz computer running Windows ME and with the DriveWire plugged into the serial port set to COM 1. The only problem a user might run into is a newer PC without serial ports or a floppy drive. This is a great reason to put that old PC back into service!

The user manual covers installation fairly well. Install the server software on the PC using the included setup routine. The server has settings for COM port setting (COM 1-6) and baud rate (38400 for the CoCo 1/2 and 57600 bps for the CoCo 3). You need to go into Windows' hardware profiles and change the default COM port setting. They're usually set to 9600 bps by default. This hung me up for a bit when I first installed DriveWire, resulting in constant I/O errors. I haven't used a COM port in, oh, years and years. Once set correctly, though, things started working a little better.

The CoCo disk includes a wizard program that builds a custom HDB-DOS disk. Back this disk up, of course. The wizard asks a series of questions to configure the OS, including setting your floppy drives up and setting the step rate to 6ms, which is handy (no need to POKE them constantly). It also creates an EPROMable version of the DOS for those that have a burner.

For NitrOS-9, the system is even easier. You first mount the correct .dsk image in the DriveWire server. These are copied to the My Documents folder during installation, along with a second folder full of ROM images of the system. There are three .dsk images for NitrOS-9: one for the Color Computer 1/2 (Level I), one for the CoCo 3 6809 (Level II), and one for the CoCo 3 6309 (Level II). You then execute a corresponding loader program on the CoCo floppy disk and NitrOS-9 loads right up from the server. This is way faster than doing it on a floppy disk. Having it boot off a ROM would make it virtually transparent.

My initial foray with DriveWire met with a little resistance. Constant I/O errors plagued the operation, even after I managed to get the COM port settings correct. In some cases, the PC and CoCo would both completely lock up. These errors were most likely caused by my flaky Windows ME installation. It likes to pause randomly and think for awhile before returning control. A quick email to Boisy Pitre at Cloud-9 fixed the problem right up. (Allow me to insert a plug here. Cloud-9's support for their products has always been top notch in my experience. Buy and install with no fear!)

The fix is to download a newer version of the server off of Cloud-9's website. Simply copy the downloaded exe file over the old one. The newer version looks better. It has photographic images of disk drives instead of a gray dialog box (see screen shot). It also runs much more stable.

## Disk BASIC Operation

First, let's get some of the technical details out of the way:

Included with DriveWire is a special edition of HDB-DOS (HBD-DOS DW), an enhancement for Disk Extended Color BASIC that lets the CoCo access hard drives. The included edition works only with DriveWire. It can't access IDE or SCSI drives hooked up to the CoCo at the same time. (NitrOS-9 can, however. See below.) HDB-DOS DW adds several commands to DECB that control access to DriveWire, as well some command enhancements and a nifty command line utility. For simplicity, when referring to HDB-DOS in this review, I'm referring to the DriveWire edition, unless specified otherwise.

Users of HDB-DOS 1.1B should be aware that the DW edition is a different version and you'll need to use the

1.1B version to access your hard drives separately. So to transfer software from the PC to your CoCo's hard drive, using DECB, you would need to first use HDB-DOS DW to transfer the file to a floppy. Then you could restart your CoCo, fire up HDB-DOS 1.1B, and copy the file to your hard drive. As Boisy Petre explained it via email, this is a limit of the DECB environment. Unlike NitrOS-9, you can only use one controller type at a time.

HDB-DOS runs the CoCo 3 in high speed mode with no problems accessing the floppies. However, if a program POKEs the CoCo back down to .87 Mhz you'll need to POKE it back up before accessing the DriveWire server.

HDB-DOS can be burned to an EPROM to replace the existing DECB ROM in your controller. An image for this is created on the boot floppy when you run the wizard program. This feature was not tested for this review. If you end up using DriveWire as your primary mass storage device, and you have access to a burner, though, this would definitely be the way to go. Cloud-9 can also provide burned EPROMs.

DriveWire acts like a hard drive, or perhaps better stated, a whole lot of floppy drives attached to your CoCo. There are four drive slots in the server, and each slot holds a dsk image that can contain up to 255 virtual disks. Access to the drives is done through the DRIVE #n command, where n is the drive you wish to access.

Loading and saving programs to and from the CoCo is as easy as using the already familiar DECB commands. SAVE, SAVEM, LOAD, LOADM, BACKUP, COPY, DSKINI, etc. all work as they should on the DriveWire virtual disks.

HDB-DOS adds a couple of commands and modifies a few of the old ones. For example, the COPY command no longer needs the second file name to work. It just uses the same file name for the copy.

Another handy command is RENAME which changes the disk label. This is particularly useful when you start filling a virtual drive with disks and want to group them together. The disk labels can be long and are not limited to 8.3 file naming. A utility that would let the CoCo display all the disks on a virtual drive, listing their number and assigned label, would be a nice addition, but alas is not included at this time.

HDB-DOS also includes the FlexiKey utility that makes command line editing a lot easier, particularly because it can recall the last few commands entered. I find myself frequently booting up HDB-DOS just for FlexiKey!

There's plenty more, including an update to the DOS command, a fix for DSKINI (no longer erases a program in memory!), and an auto execute feature.

The HDB-DOS DriveWire edition is pretty much the same as the regular edition of HDB-DOS, but does not include several utilities. A full review of HSB-DOS is beyond the scope of this review, but it certainly worth considering if you're interested in working with DECB and a hard drive. You can view the entire manual for the regular edition of HDB-DOS at www.cloud9tech.com.

## NitrOS-9 Operation

As said, using DriveWire with NitrOS-9 is simple. Booting is accomplished from an included .dsk image that contains the operating system. Just put the boot disk in the CoCo's floppy and execute the correct loader. NitrOS-9 boots from the DriveWire server automatically.

You can use a real hard drive in conjunction with DriveWire under NitrOS-9. NitrOS-9 can be configured to assign different drive designations to different devices. So you could, for example, have an IDE hard drive, a SCSI CD-ROM, two floppies and DriveWire all hooked up and usable at the same time.

Not being terribly familiar with NitrOS-9, I wasn't able to test everything. However, I was able to move files around, transfer them to floppy, open files, and execute programs via the DriveWire without any difficulty. If you're interested in NitrOS-9, the ability to use it this way is worth the price of the DriveWire package alone. The OS really shines when it has a little elbow room to play with!

Shuffling Back and Forth

Until you get things organized, there is a bit of back forth from the CoCo to the PC as you switch disk images. This can be a blessing, though, for getting your CoCo's software collection organized. For example, you can create libraries of like programs and files in separate disk images and mount them as needed. The CoCo can't use long file names, but the file names of dsk images on the PC can. Each CoCo disk can also be labeled using the RENAME command in HBD-DOS. Loading multi-disk programs seems to require going back and forth, however. But once you've got everything loaded over to the PC in dsk format, you might find there's little need to attach floppy drives to your CoCo!

Everything I loaded from the PC fired up just fine. I did not try to load any programs that needed to be cracked because of disk protection. These may pose a problem for DriveWire. Those that were already cracked worked fine. I also didn't fire up anything using the DOS command (nothing on the PC that used it). If it doesn't work I would suggest the old Radio Shack DECB 1.0 loader could be used.

The primary reason I bought DriveWire, however, was to transfer files from my PC to the CoCo and vice versa. Particularly, I wanted an easy way to get stuff downloaded from the Internet on to a real CoCo floppy. If you're in a similar frame of mind, DriveWire fits the bill nicely, without having to install an old 5 1/4" in your PC or mess with old terminal software.

Note: In email, Boisy mentioned that the DriveWire cable is wired like a standard NULL modem cable, so you could potentially use it with terminal software on both ends to transfer files back and forth, if you were inclined to do so. This feature would be particularly handy for transferring text, pictures, music, or other data that you don't want to have to convert to .dsk first. It also opens up the possibility of creating other software that uses the cable for communications between the CoCo and the PC. An automated transfer utility running in the background under NitrOS-9 might be interesting, for example.

To illustrate how easy it is to transfer things back and forth, here's a quick look. HDB-DOS uses the DRIVE command to designate which of the four (0-3) drive slots the server will use. Each of these virtual drives, however, can hold 255 virtual disks. For the example to follow, I'll use the terms "virtual drive" to indicate the server's drive slot, and "virtual disk" to indicate the disk on that virtual drive. The reason for this is that the DRIVE command, with the OFF/ON switch, can also indicate which virtual disk your CoCo is accessing. You need to turn the virtual disks off to access the real disks in your CoCo's floppy drive. This is, of course, a limitation of DECB, but the way Cloud-9 got around it is clever. It works like this:

Let's say we want to transfer a dsk image of a game to a real CoCo floppy disk.

**Step 1**. Type DRIVE #0 on your CoCo. This sets the server to access the virtual drive 0 on the PC. (You could just as easily accomplish the same tasks on Drives 1, 2, or 3 by typing DRIVE #n, where n is the drive you want to work with.)

**Step 2.** Load the dsk image on the PC into the first (bottom) virtual drive on the server.

**Step 3.** Type BACKUP 0 TO 1. This copies all the files on virtual disk 0 to virtual disk 1 on the selected virtual drive. Once a dsk image is mounted, it has 255 eligible virtual disks available. DriveWire handles this transparently. There's no need to "create" these disks. Once an image is mounted in a virtual drive, it can hold as much as you can cram into its individual virtual disks.

**Step 4.** Type DRIVE OFF 0 on the CoCo. This turns off virtual disk 0 on the selected virtual drive. This in turn allows access to the real disk 0 on the CoCo's floppy drives. DRIVE OFF n turns off all virtual disks on the active virtual drive equal to or lower than the number specified. Real CoCo disks with those numbers can now be accessed. For example, DRIVE OFF 2 turns off virtual disks 0,1, and 2. You can also specify a range of disks.

**Step 5.** Type BACKUP 1 TO 0. This will copy the files in virtual disk 0 on the server to the real disk drive 0 attached to the CoCo. This is why Step 3 above is necessary. Once virtual disk 0 is turned off, to allow access to the floppy drives, you can't access it any more. So, all the files need to be copied to a virtual disk with a drive number higher than the real floppy drive you'll be using.

Step 6. Type DRIVE ON to disable the real floppy drives and reactivate all the DriveWire disks.

That's it. Instant floppy disk you can now use on your CoCo. Copying files to the PC from the CoCo is as simple as reversing the procedure. Just DRIVE OFF 0, BACKUP 0 TO 1, DRIVE ON, BACKUP 1 TO 0.

(It's a good idea to copy the files back to disk 0, since an emulator and other PC/CoCo utilities might not see them otherwise.)

This may seem like a bit of work, just to get a working CoCo floppy disk, but it's quite intuitive once you do it a few times. The scheme also allows for a great deal of compatibility with regular DECB. There's no new file system for use with the drives on your PC. DriveWire just uses the familiar DECB system. You can also use programs that don't let you specify which disk to use, for example games that automatically save to drive 0.

## Is It Worth It? Oh, Yeah!

DriveWire does have its limitations. From talking with some CoCo users, I've learned that it's not as fast as a real hard drive. If you need blazing speed, a SCSI drive and controller is probably the way to go. On the other hand, DriveWire is considerably faster than a floppy. For most users, DriveWire will be plenty fast enough.

Even with the newer version in place, DriveWire server still locks up occasionally on my PC. As mentioned, it's a very old installation of Windows ME. Most of the problem is there. The lock ups occur when I'm doing something else on the PC and letting the server run in the background. Basically, it doesn't appear to be very tolerant of interruptions. For the most part this isn't a problem, but it's something a user will want to test, and compensate for, before trying anything mission critical.

Overall, DriveWire is brilliant. Installation and operation is easy and intuitive. For a mere US$40.00 (at the time of this writing), plus shipping, you can add mass storage to your Color Computer. This is, hands down, the easiest way to transfer files between your PC and CoCo. If you use DECB, the HDB-DOS enhancements are a great bonus by themselves. NitrOS-9 users get another great storage option and the ability to transfer and execute programs quickly. In short, DriveWire makes any CoCo even better and more capable than ever.

## Links

DriveWire is available for order from Cloud-9 Tech at:
www.cloud9tech.com

Please see the website for the latest pricing and options available.